The Dissertation Committee for Alvaro Enrique Barrera certifies that this is the approved version of the following dissertation:

History Matching by Simultaneous Calibration of Flow Functions

Committee:

Sanjay Srinivasan, Supervisor

Steven L. Bryant

A. Stan Cullick

Larry W. Lake

Gary A. Pope

History Matching by Simultaneous Calibration of Flow Functions

by

Alvaro Enrique Barrera, B.S.; M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December 2007

Acknowledgements

I would like to express sincere gratitude to my supervisor and mentor, Dr. Sanjay Srinivasan. Dr. Srinivasan's expert advice and creative insights have had a profound and undeniable impact upon this dissertation. More importantly, my career and personal life have benefited greatly from Dr. Srinivasan's ongoing support and honest guidance during my graduate studies.

I would also like to thank Steven Bryant, Stan Cullick, Larry Lake and Gary Pope for their commitment of time, as well as their constructive observations, while serving on my dissertation committee. I would also like to thank these gentlemen for their scholastic contributions not only to this dissertation but also to my future career in Petroleum Engineering.

To my fellow graduate research assistants, Shinta Reinlie, Juliana Leung, Kiomars Eskandari, Yonghwee Kim, Louis Forster, Cesar Mantilla and Aviral Sharma, I thank you for your unfailing assistance and loyal friendship during my studies at The University of Texas at Austin.

History Matching by Simultaneous Calibration of Flow Functions

Publication No._____

Alvaro Enrique Barrera, Ph.D.

The University of Texas at Austin, 2007

Supervisor: Sanjay Srinivasan

Reliable predictions of reservoir flow response corresponding to various recovery schemes require a realistic geological model of heterogeneity and an understanding of its relationship with the flow properties. This dissertation presents results on the implementation of a novel approach for the integration of dynamic data into reservoir models that combines stochastic techniques for simultaneous calibration of geological models and multiphase flow functions associated with pore-level spatial representations of porous media.

In this probabilistic approach, a stochastic simulator is used to model the spatial distribution of a discrete number of rock types identified by rock/connectivity indexes (CIs). Each CI corresponds to a particular pore network structure with a

characteristic connectivity. Primary drainage and imbibition displacement processes are modeled on the 3-D pore networks to generate multiphase flow functions corresponding to networks with different CIs. During history matching, the stochastic simulator perturbs the spatial distribution of the CIs to match the simulated pressures and flow rates to historic data, while preserving the geological model of heterogeneity. This goal is accomplished by applying a probabilistic approach for gradual deformation of spatial distribution of rock types characterized by different CIs. Perturbation of the CIs in turn results in the update of all the flow functions including the effective permeability, porosity of the rock, the relative permeabilities and capillary pressure.

The convergence rate of the proposed method is comparable to other current techniques with the distinction of enabling consistent updates to all the flow functions. The resultant models are geologically consistent in terms of all the flow functions, and consequently, predictions obtained using these models are likely to be more accurate. To compare and contrast this comprehensive approach to reservoir modeling against other approaches that rely on modeling and perturbing only the permeability field, a realistic case study is presented with implementation of both approaches. Comparison is made with the history-matched model obtained only by perturbing permeability. It is argued that reliable predictions of future production can only be made when the entire suite of flow functions is consistent with the real reservoir.

Table of Contents

List of Tables

List of Figures

1 INTRODUCTION

A two steps process is generally followed to develop models of reservoir heterogeneity. First, a geological model is obtained by the integration of static geological information from different sources such as seismic, well logging and sequence stratigraphy information. Then, during history matching (HM) or dynamic data integration, the model is adjusted or modified to match the production history.

Reproduction of the historical production data is usually the most important objective during the validation of a reservoir model. Reconciling the difference between the model prediction and field production records, however, represents a rather significant challenge, considering the highly non-linear relationship between the reservoir attributes and the production response.

The final purpose of every reservoir model is to provide reliable predictions of reservoir performance. Although the accuracy of a predictive model can not be easily evaluated or measured before the actual event occurs, it seems reasonable that models with a more accurate description of the reservoir heterogeneity would produce more realistic estimates. It is hypothesized in this dissertation that the process of calibration

of reservoir models with dynamic data renders the reservoir models more consistent, and consequently, predictions of future reservoir response are likely to be more robust.

1.1 PROBLEM STATEMENT

There are two common practices for matching the response of the reservoir model with the production history. In the first approach, HM is limited to the calibration of macroscopic static reservoir attributes ignoring the multiphase flow functions. The second frequent approach for history matching is the calibration of the multiphase functions with a fixed geological model. The drawbacks of both these practices are that they rely on the calibration of either the geological model or the multiphase flow functions, assuming independency of one from the other.

This research consolidates both approaches for history matching in a single method that is based on a consistent depiction of lithologic variations. All the flow properties of the rocks, e.g. relative permeability, porosity, permeability, etc., are consistently perturbed. The relationship between all properties of the rock can be captured at the pore level where the intrinsic parameters of a particular geological setting, e.g. grain size distribution, grain scale spatial correlation, play an important role on the driving mechanisms for multiphase flow.

1.2 OBJECTIVES

This dissertation focuses on a probabilistic approach to integrate dynamic data with reservoir models that improves the consistency between multiphase flow functions and geological heterogeneity. At the field level, the algorithm relies on an efficient parameterization of the dynamic data integration problem and permits rapid updating of the reservoir model using production data available at each stage of reservoir development. At the pore level, the algorithm relies on a realistic model for the fundamental flow mechanisms and the corresponding multiphase flow functions based on an appropriate representation of the pore-space.

The general objectives of the research include:

- Development of a method to calibrate multiphase flow functions using pore network models that are characteristic of a rock type, identified by a particular connectivity index.

- Implement an optimization framework for assisted history matching based on stochastic perturbation of connectivity indexes throughout the reservoir domain. This yields reservoir models that exhibit consistent variations of all properties (static as well as flow).

1.3 APPROACH OVERVIEW

A stochastic simulator is used to generate the geological model, represented by the spatial distribution of connectivity indexes (CIs), corresponding to different rock textures or types. Each rock texture, identified by a CI, is described with a characteristic pore-space connectivity. The difference in pore connectivity between the different rock textures in turn depends on sedimentary processes. Different three-dimensional pore networks, based on the representation of the pore space for each rock texture and its characteristic connectivity, are generated. Primary drainage and imbibition displacements are modeled on these pore networks to generate multiphase flow functions linked to each rock texture or CI. During the assisted history matching, the stochastic simulator perturbs the spatial distribution of the CIs to match the simulated pressures and flow rates to historic data. Perturbation of the CIs in turn results in the update of all the flow functions and their spatial allocation in the reservoir model. Flow functions obtained from the pore-network models include the effective permeability and porosity of the rock, the relative permeabilities and capillary pressure. Since the resolution of the flow simulation model is much coarser than the scale over which lithofacies variations are observed, a study investigating the scale up of flow functions representing the properties of a support volume that contains a mixture of facies, is also presented. Figure 1-1 shows a schematic of the

integrated approach for the simultaneous calibration of geological models and multiphase flow functions.

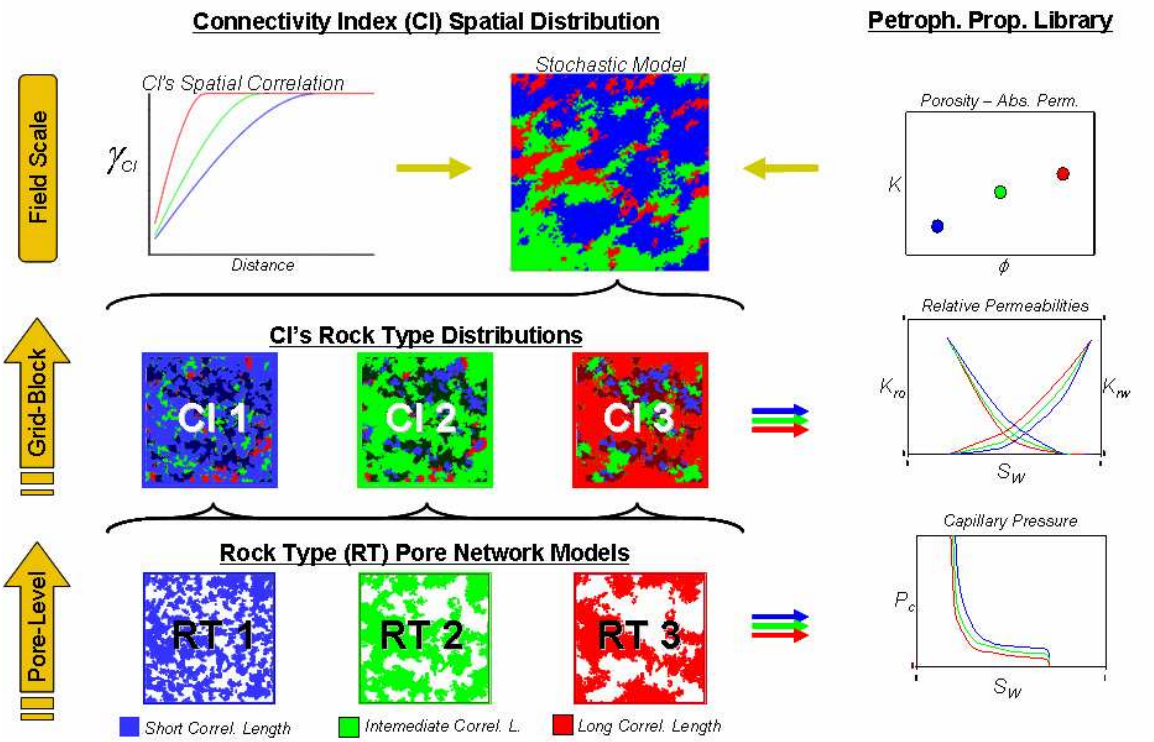


Figure 1-1. Proposed approach for the generation of reservoir model with geologically consistent multiphase flow functions.

An algorithm that uses a probability perturbation method for gradual deformation of geological models in a history matching context has been implemented. This approach, compared to other perturbation methods, offers the important advantages of preserving the prior geological heterogeneity model and reducing the set of parameters to be calibrated during a history match. The algorithm

couples the sequential indicator simulation algorithm for geological modeling with a flow simulator in a Markov-Chain approach where the reservoir model is iteratively updated. The local conditional distributions of rock type are progressively updated using a deformation parameter that is calibrated using the production information. Therefore, at convergence this dynamic parameter quantifies the information in the production data pertaining to the reservoir heterogeneity. Local distributions conditioned to static and dynamic information are iteratively updated using the dynamic parameter until a global match to the historic data is attained.

1.4 DISSERTATION OUTLINE

This dissertation is organized into nine chapters. A short literature review of the past work done in the areas of history matching, pore structure representation and pore network modeling, is presented in Chapter 2. The following chapter presents a more detailed description of the modeling framework, including the workflow and remarks concerning the structure and interaction of the main components. A description of the probability perturbation approach for history matching reservoir models at field scale and the results for a synthetic case are presented in the Chapter 4. Chapter 5 presents the details of the pore network modeling work, including the method for construction of pore network models consistent with the geology, the modeling of invasion processes and the computation of multiphase flow functions.

The influence of pore-level spatial connectivity on capillary pressure and relative permeability curves is demonstrated. Results and observations from preliminary evaluations and sensitivity studies of multiphase flow at pore level are summarized in the Chapter 6. In Chapter 7, the application of the multi-scale approach for history matching reservoir models by calibrating the static model along with multiphase flow functions is presented for a synthetic case. The results and observations for a field scale application of the proposed integrated history matching approach are presented in Chapter 8. Finally, a summary of the results, observations and conclusions based on the application of the proposed methods is presented in the Chapter 9.

2 REVIEW OF RELEVANT LITERATURE

2.1 DYNAMIC DATA INTEGRATION

The integration of dynamic data into reservoir models is significantly more challenging than integration of static data (porosity, permeability), mainly because of the highly non-linear relationship between dynamic data and the model variables. Additionally, reservoir properties are variable even at small scales, demanding models with grids that are as fine as computationally feasible, translating into a large number of model parameters. Simultaneous optimization of the attribute value in all these grid blocks is therefore a computational challenge. All these factors make traditional approaches such as Markov chain Monte Carlo and genetic algorithms, inappropriate and impractical for most field problems.

The most common approach for the process of matching the response of the reservoir model with the production history is the calibration of geological models at the macroscopic and structural levels ignoring the multiphase flow functions. Multiple methods available to condition reservoir models to dynamic flow data have

been developed (Yeh, 1986; de Marsily et al., 1995; Gomez-Hernandez et al., 1997; Omre and Tjelmeland, 1996; Roggero and Hu, 1998).

Another common approach for history matching in the industry is the calibration of the multiphase functions with a fix geological model (Watson, 1984; Labban and Horne, 1991; Tan, 1995; Denbina et al., 1998; Christie et al., 2002). Relative permeability and capillary pressure curves are important elements in the assessment and prediction of the recovery in oil reservoirs. Representative curves are normally obtained through special core analysis laboratory (SCAL) studies, by reproducing as close as possible the reservoir conditions. The laboratory measured "rock" curves are adjusted in the simulation model during the history matching step, largely relying on the reservoir engineering expertise of the reservoir modeler. This procedure has been demonstrated to be effective in a large number of history matching cases.

The drawbacks of both these practices are that they rely on the calibration of either the geological model or the multiphase flow functions, assuming complete independence of one from the other.

2.1.1 CALIBRATION OF THE STATIC MODEL

The Gauss-Newton method can be efficient for history matching problems with a large number of variables but a small number of data (or alternatively a large number of data that are highly redundant) (Tarantola, 1987). These gradient-based optimization techniques use sensitivity coefficients (Chu et al., 1995) of the flow response to the attribute being modeled. Calibration of sensitivity coefficients requires solution of the flow equations using a reservoir simulator. Consequently, they are extremely time consuming and require a large number of flow simulation runs. Gradient-based methods have been used for solving large unconstrained minimization problems; however, history matching problems with a large number of model parameters and production data are significantly more complicated.

Iterative methods, such as conjugate gradient and steepest decent, use mathematical approximations instead of local gradients or sensibilities to compute search directions during the history matching process (Jahns, 1966; Chen et al., 1974). These methods are characterized by relatively low computational requirements for each iteration, but a large number of iterations, particularly for multiphase flow (Makhlouf et al. 1993). Quasi-Newton methods, which use a numerical approximation of the Hessian, show an important improvement in the convergence characteristics. Yang and Watson (1988) and Deschamp et al. (1998) concluded that

quasi-Newton methods are considerably more efficient for large field problems, while steepest descent-Gauss-Newton hybrid methods are more suitable for relatively small problems.

In the reparameterization approach a reduction in the number of model parameters can be obtained by writing the parameter fields as linear combinations of a relatively small set of basis vectors. In some cases, under this approach, the Gauss-Newton method can be used efficiently for large problems. In general a poor choice for the initial subspace vectors may result in a slow convergence. The implementation of the reparameterization methods requires the efficient computation of gradients of the objective function with respect to the basis vector parameters. The simplest example of the reparameterization method is history matching by zonation, where domains larger than a single simulation grid block are defined. This concept was used in some of the first automatic history matching studies (Jacquard and Jain, 1965; Jahns, 1966), and is frequently used when attempting a history match by manually adjusting reservoir parameters. Bissel et al. (1994) proposed a zonation based on sensitivity to data. An important drawback of the zonation method is the presence of undesirable discontinuities in the reservoir properties at the boundaries of the zones.

Another reparameterization approach, the pilot point method was introduced by deMarsily et al. (1984). This method and the closely related master point method have been extensively used in groundwater problems (Lavenue and Pickens, 1992)

and became one of the preferred reparameterization approaches for history matching. In this approach the parameter space is reduced to a limited number of locations (pilot points) conveniently distributed throughout the reservoir model domain. The value of the attribute at the limited number of pilot points is optimized during the history matching process, while the attribute values in the rest of the reservoir model are interpolated based on a prior covariance model. However, presence of undesirable artifacts at the pilot point locations (Landa and Horne, 1997), refuted the common belief that the method yielded models that were geologically consistent. Other reparameterization methods include the use of eigenvectors (Gavalas et al., 1976, Oliver, 1996) and the use of spline functions (Distefano and Rath, 1975; Lee and Seinfeld, 1987).

In the gradual deformation method (Roggero and Hu, 1998), some times considered a reparameterization method for conditional simulation, the solution is searched in a subspace spanned by a small number of realizations. This recently popular approach allows a controlled and smooth transition of the reservoir model to improve the match of the production data while preserving structural constrains. Initially proposed for modeling multi-Gaussian fields, the gradual deformation approach has been extended to non-Gaussian fields at the cost of increased computational effort.

In a different approach, the use of fast approximate simulators allows the computation time to be reduced. The reduction in computational time with fast simulators such as coarse gridded simulators, streamline simulators (Datta-Gupta et al., 1995; Bratvedt et al., 1996) and neural networks, allows the application of methods that would otherwise be impractical with traditional simulators. Frandsen et al. (1996) proposed using fast approximate mathematical models, also know as proxy or surrogate models, to minimize the objective function during history matching. These mathematical models are built and periodically recalibrated using the true (traditional) simulator.

Other approaches include sequential self-calibration methods (Gomez-Hernandez et al., 1997), Markov chain Monte Carlo methods (Omre and Tjelmeland, 1996) and ensemble Kalman Filter (Naevdal, 2003).

Caers (2002), and Kashib and Srinivasan (2003) proposed a history matching approach where the conditional probability distributions of static properties (porosity, permeability), are systematically perturbed to match the dynamic data while preserving the geological structure. In this approach multiple realization of the reservoir model can be sampled from the updated probability distributions. Integration of the dynamic data with the geologic structural information to obtain the updated conditional probability distributions is obtained using the permanence of ratio hypothesis (Journel, 2002). In this research, the probability perturbation

approach is extended to model rock type indexes, such that the converged, history matched model reflects all the rock properties (static as well as flow functions) accurately.

In spite of the intense research activity in this area and significant achievements during the last decade, the number of relevant realistic field applications of assisted history matching is still rather limited; given the complexity of the problem, there are still several issues to be resolved.

2.1.2 CALIBRATION OF MULTIPHASE FLOW FUNCTIONS

A complete and accurate set of multiphase flow functions is required to properly characterize the rock-fluid interactions in the most relevant lithologies of a reservoir. Incomplete knowledge about the flow processes occurring at the pore level, as well as the uncertainty linked to the process of upscaling experimental flow functions from laboratory scale to volumes representative of grid blocks in the simulation model, render the process of calibration of these flow functions using multiphase production data, a crucial step in the process for predicting reservoir performance. A frequent practice in this calibration approach is the use of constant multipliers to scale parameters of the multiphase flow functions field-wide, in particular regions or lithologies. The calibration of the multiphase flow functions is

normally accomplished by adjusting one or more parameters, including the initial or connate water saturation, the oil residual saturation, the relative permeability end points for the present phases and finally the curvature of the function.

The calibration of multiphase flow function during the history matching process was initially performed manually by trial and error. Archer and Wong (1973) were the first researchers who considered the estimation of relative permeability curves by using a reservoir simulator to match laboratory core flood data. They estimated parameters that defined the shape of simple empirical relative permeability models by trial and error. Sigmund and McCaffery (1979) proposed for the first time, the use of nonlinear regression to match core flood data with power law models of relative permeability curves. Kerig and Watson (1986) considered using cubic splines to parameterize the relative permeability curves. Lee and Seinfeld (1987) considered the simultaneous calibration of relative permeabilities along with the absolute permeability in a two-dimensional, two-phase oil-water system assuming power law relative permeability curves and known end points.

Yang and Watson (1991) considered the estimation of relative permeabilities from production data using a Bayesian approach, modeling the relative permeability functions as linear combination of B-splines. Watson et al. (1980) considered the simultaneous estimation of porosity, absolute and relative permeabilities by automatic history matching of production data, but the method was restricted to two-

dimensional oil water systems with homogeneous permeability and porosity. Kulkarni and Datta-Gupta (2000) proposed the estimation of relative permeabilities from water cut and pressure data for an oil-water system using automatic history matching with a stream line simulator, approximating the relative permeability curves with both b-splines and power law models.

In the proposed approach the multiphase flow functions are not constrained by a mathematical model. Instead, they are estimated based on rock-fluid interactions defined at pore level and then upscaled based on a geological heterogeneity model. Rock-fluid interactions are determined using invasion algorithms in network models that represent pore structure.

2.2 PORE SPACE STRUCTURE

Microscopic pore-level properties and spatial correlations have an important role in the macroscopic characteristics of multiphase flow, such as absolute permeability, tortuosity, residual saturation, capillary pressure, and relative permeability curves. Although a significant number of studies (Mohanty and Salter, 1982; Lenormand et al., 1983; Chatzis and Dullien, 1985; Wardlaw et al., 1987; Celia et al., 1995) have investigated the role of the pore structure on fluid transport through

porous media, most of them assume simple descriptions of the pore structure and/or exclude part of the underlying physics of the fluid displacement mechanisms.

More recent studies have demonstrated that the spatial distribution and correlation of pore level geometrical properties would also influence the bulk multiphase flow properties (Tsakiroglou and Payatakes, 1991; Ioannidis et al., 1993, Matthews et al., 1995; Bryant et al., 1995; Rajaram et al., 1997). In multiphase flow at a pore scale, capillary forces determine the distribution of flowing phases, controlling in turn the fundamental behavior of multiphase flow at a continuous scale. Consequently, a reasonable description of the pore space is required to better comprehend the fundamental role of pore scale structure on the macroscopic characteristics of the multiphase flow through porous media.

The seemingly variable characteristics of the rock at a microscopic scale require sophisticated analysis of geometric parameters and their spatial correlation. Thus, early porous media models focused on bulk average quantities such as porosity, specific surface area, formation factor and tortuosity, while making simplistic assumptions about the detailed structural description of the pore space.

In a large number of pore network models developed before 1990's, the geometrical properties of the pore structure were estimated based on capillary pressure-saturation curves from mercury porosimetry studies (Wardlaw and Taylor,

1976; Dullien, 1979; Gueguen and Palciauskas, 1994; Tiab and Donaldson, 1996). However, that was an indirect approach that captures primarily the statistics of the largest pores only. Furthermore, the procedure does not yield any information on the spatial correlations of the properties. Pore casting (Wardlaw, 1976) and Wood's metal Porosimetry (Dullien, 1981) can be used for the direct visualization of the pore space; however, these techniques also have very limited application due to the difficulty to obtain quantitative measurements from the casts and the destructive nature of the method.

Reconstructed models of porous media have been used to characterize the pore space geometry and evaluate its influence on the macroscopic flow properties (Quiblier, 1984; Adler et al., 1990; Giona et al., 1996). A particular approach is the reconstruction of porous media by modeling the basic geological processes (Bryant and Blunt, 1992; Bakke and Oren, 1997; Pilotti, 2000). Numerical models for sedimentation, compaction, and diagenesis processes were used to create pore structure models (Bakke and Oren, 1997). Models obtained by the numerical reconstruction of the pore space permit evaluation of transport and mechanical processes. The process-based method is general and allows the reproduction of the long range pore connectivity, particularly in clastic reservoirs. However, the approach is still limited to simple grain geometries and has significant limitations for reproducing the characteristics of carbonate rocks. This is due to the complexity of

the geological processes of compaction, dissolution, and chemical reactions that have a significant impact on the characteristics of carbonate rocks.

Direct measurements of pore structure properties were largely restricted to the analysis of thin sections and serial stacks of thin sections until the 90's (Lin and Cohen, 1982; Koplik et al., 1984; Doyen, 1988; Lymberopoulous and Payatakes, 1992). Computerized image analysis and pore mapping software for thin sections was introduced by Dullien and collaborators (Yanuka et al., 1984; MacDonald et al., 1986; Kwiecien et al., 1990). Limitations of such approaches include the destructive nature of the technique (during the process of obtaining thin sections), errors introduced by the difference between the thickness of the thin section and the resolution of the digitization process. The method is also incapable of measuring non-isotropic pore properties, and besides, requires extensive work to polish, slice, and digitize the samples. The advantages, on the other hand, include the ability to measure properties such as specific surface area and porosity.

Statistical techniques also allow the generation of 3 Dimensional pore space representations. Two-point or multiple-point statistics can be used for these representations (Quiblier, 1984; Adler et al., 1990, 1992; Roberts, 1997; Roberts and Torquato, 1999; Yeong and Torquato, 1998; Manwart et al., 2000, Talukdar et al., 2002; Caers, 2001; Strebelle et al., 2003). N-point covariance models were introduced to characterize the spatial distribution of pore space properties and their impact on the

bulk medium properties (Berryman and Blair, 1986; Torquato, 1991; Thovert et al., 1993; Coker and Torquato, 1995). The full set of n-point covariance functions provides a complete description of the pore space geometry. Computational requirements yet limit the estimation of the n-point covariance functions to only the lower order (n<=3).

The introduction of the synchrotron X-ray computed microtomography (Flannery et al., 1987; Spanne and Rivers, 1987; Deckman et al., 1989-91; Kenney and Nichols, 1989) and laser scanning confocal microscopy (Fredrich et al., 1995) brought a significant improvement in the direct measurement of pore space properties. Both methods are non-destructive, offering uniform resolution in three dimensions, and requiring less time/work effort compared to the aforementioned techniques. Different software for the geometric analysis of digital data sets obtained using these techniques have been developed (Lindquist et al., 1996-99). Even though laser scanning confocal microscopy offers a higher resolution, it has limited applicability on thick objects. This last technique is based on the imaging of fluorescent dye injected in the samples.

2.3 PORE NETWORK MODELS

Empirical network models of pore structure allow the calculation of petrophysical properties without relying on laboratory measurements. They represent a practical alternative to experimental data, which is inherently difficult to obtain and only found for a small variety of rock types. Pore network modeling involves the idealized representation of the pore lattice connected by throats to determine critical multiphase transport properties.

Pore network models have been used to study single and multiphase flow through porous media. Early studies employed percolation concepts, which were limited to simple pore space descriptions and exclude some of the physics of the actual fluid displacement process. Later on, more elaborated network models were used to evaluate the effect of pore-level spatial correlations in macroscopic properties of porous media (Mohanty and Salter, 1982; Jerauld and Salter, 1990; Blunt and King, 1991). However, only a limited number of these studies have addressed the impact of pore-space correlations on both imbibition and drainage displacement mechanisms (Mani and Mohanty, 1999). Moreover, none of those studies considered the impact of multiphase flow functions on field-scale flow simulation.

Network models and lattice Boltzmann methods have been used to assess the relationship between geological heterogeneity at the pore level and the bulk properties of the rock, including the multiphase flow functions (see review by Celia et al., 1995). Lattice Botlzmann methods require a digital representation of a sample, while pore network models rely on an effective description of the pore space. As a result, larger rock volumes can be modeled using pore network models, considering the same computational power.

Originally, multiphase flow in porous media was modeled by analogy to fluid behavior in bundles of capillary tubes (Childs and Collins-George, 1950; Burdine, 1953; Mualem, 1976). These initial models ignored the natural topological complexity in the connectivity of the pore space, leading to simplistic models for relative permeability and saturation. A two dimensional network of capillary tubes with a random distribution of radius was originally proposed by Fatt (1956), one of the pioneers in the development of pore network models. This original model generally considered capillary tubes that were connected directly to each other. More recent models (Celia, 1995; Ioannidis and Chatzis, 1993; Dullien, 1992; Wong and Zhu, 1999), however, consider junctions with an effective volume representing the larger void spaces, commonly known as pore bodies or sites. The capillary tubes, representing the narrow apertures that connect the pore bodies, are known as throats or bonds. Pore bodies and throats are sampled from representative size distributions, where the pore bodies are normally modeled as spheres, while the throats are

represented with cylindrical or conical shapes. At least two different throat parameters are needed to arrive at a realistic pore space description, the throat size or radii, and the throat length. Both parameters play important roles in the relationship between the pore space geometry and the multiphase flow functions (Mohanty and Salter, 1982; Chatzis et al., 1983; Lenormand et al., 1983; Chatzis and Dullien, 1985; Wardlaw et al., 1987, Celia et al., 1995).

Another important topological descriptor of the pore network is the connectivity –the number of throats associated with a single pore body or coordination number. Initial models considered a uniform connectivity (regular lattices of 4 and 6 being the most common) –Mayagoitia et al., 1988; Celia, 1995; Dullien, 1992; Bakke and Oren, 1997. More recent and improved models consider the spatial variation of the coordination number and its correlation with other pore-scale properties.

More recent advances in pore scale imaging techniques using microtomography (pioneered by Bryant and Blunt, 1992, refined by Oren, 1997, 2002) allowed the construction of more realistic lattices of pores and throats. Pore network simulators have become increasingly popular analytical tools for determining multiphase transport properties because experimental data on a variety of rock types have been successfully reproduced by pore network simulations. Water-wet sandstone models have been validated against two-phase relative permeability data gathered on

core samples (Blunt et al., 2001; Patzek, 2001). Saturation functions obtained by core-flood of carbonate samples (Yortsos et al., 1999; Kamath et al., 1998) have been accurately predicted by pore network simulations. Immiscible displacements in fractured media can also be modeled using pore-networks as proven by laboratory experiments (Glass et al., 1998; Amundsen et al., 1999). The network models can be calibrated with available experimental data to become predictive tools for determining the relative permeabilities and capillary pressures curves.

3 MODELING FRAMEWORK

The proposed approach for history matching using geologically consistent multiphase flow functions is based in the integration of an appropriate structural description of the pore space, a complete model for multiphase flow displacements, a flexible technique for geological modeling and an effective method for history matching. In this chapter, these main components of the approach are briefly introduced and linked to each other within the global scope of the dissertation.

3.1 WORKFLOW

The following scheme describes the workflow for multi-scale geological modeling and history matching proposed in this research. The approach is based on the proper characterization of the geological model at three particular scales, the pore level, the simulation-grid-block level and the field scale. Auxiliary methods for constructing pore network models from basic rock properties and scaling up multiphase flow function from pore model to simulation grid block scale are presented, to facilitate the construction of geologically consistent models.

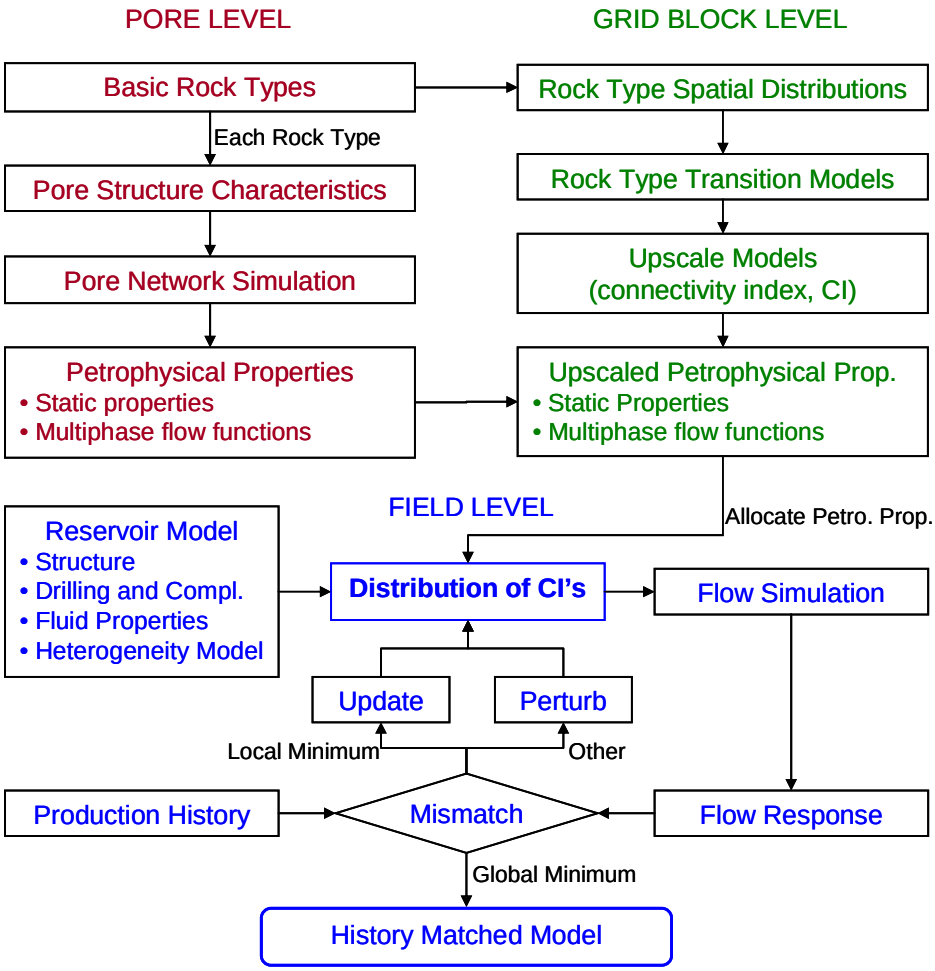


Figure 3-1. Workflow for the proposed multi-scale approach for history matching reservoir models.

3.2 PORE SCALE REPRESENTATION OF REAL LITHOLOGY

At the pore level the geological modeling process starts with the appropriate characterization of the different sedimentary units or lithologies in terms of the basic elements that constitute the pore structure, including the pore bodies and throats, their

connectivity and correlation/cross-correlation. The information required for the reconstruction of the pore structure from a rock sample can be obtained by processing the digital data from a computerized microtomography with software for geometric analysis. However, the application of microtomography techniques to study rock samples in oil fields is still rather unusual and far from been considered as part of a regular tool-kit for reservoir characterization. Considering this drawback, an auxiliary method for the description of pore structure was developed. This method considers only properties of sedimentary rocks that are more easily available, such as grain size distribution, sorting, compaction and cementation.

The proposed method for the reconstruction of pore structure is based in the analysis of microtomography studies reported in literature. In these studies, a characteristic description of the basic elements of the pore structure in sedimentary rocks is presented along with their correlation with properties that are normally measured at core laboratories.

The list of parameter required for the generation of a pore network includes the average and standard deviation of the grain size distribution, and the average porosity and cementation values. The basic property required for the construction of the pore network model is the pore body distribution. The probability distribution of pore bodies is determined from the grain size distribution and the spatial allocation is obtained using sequential Gaussian simulation with a particular correlation length

(determined by user input or correlated with the standard deviation of the grain size – Venkatarangan, 2000). Probability distributions for the other elements and properties of the pore network model, including the throat size, throat length and coordination number, are determined by the average grain size, porosity and cementation. However, the spatial distribution of these properties is determined based on a direct or inverse correlation with the pore bodies. More details about the construction of the pore network models are presented in the Chapter 5.

Pore network models representing the most relevant litho-types or rock types in the reservoir of interest, are built considering distinctive pore structures and connectivities. These pore network models are used to simulate primary drainage and imbibition displacements to determine petrophysical properties including absolute permeability, capillary pressure and relative permeability curves. The invasion algorithm developed in this dissertation considers capillary controlled oil-water displacement mechanisms. However, it can be extended to gas-liquid displacements controlled by mixed or viscous forces, according to developments presented in more recent literature (Blunt 2001).

The results of the pore network simulator with the different rock type models represent the multiphase flow library that will be used finally in the reservoir model at the field scale. Individual multiphase flow functions generated by the pore network simulator are evaluated using common understanding of multiphase flow through

porous media, and also can be calibrated with laboratory results whenever they are available. More information about the analysis and validation of the multiphase flow functions generated with pore network models is presented in Chapter 6.

3.3 SCALING UP PORE SCALE FLOW FUNCTIONS TO SIMULATION INPUTS

The estimation of petrophysical properties for reservoir models is inherently uncertain considering the small number of spatial locations with data to characterize the rock and rock-fluid properties. Additional uncertainty is associated with the extrapolation of petrophysical properties measured at laboratory or other scales, for use in large-scale simulation models. Effective properties may differ substantially with relative length scales due to the presence of nonlinear and coupled phenomena. Honarpour et al (1994) showed using the results from several displacement experiments conducted on cross and parallel bedding cores, that the structure of the heterogeneity itself played a major role in the shape of the relative permeability curve. These conclusions were confirmed by Mannseth et al. (1998) and Hamon and Roy (1998).

The nature of the geological heterogeneity in porous media at different scales and its impact in the fluid flow should be considered in the definition of the

petrophysical properties to be used in a flow simulator. The multiphase flow functions calculated from pore network models should be representative only for that scale. However in the upscaling approach presented in this dissertation, these pore level flow functions can be used as the basis for obtaining flow functions representative of larger volumes, including normal grid blocks used in flow simulators.

The proposed flow-based upscaling approach is based on the application of the steady state method to determine relative permeability from a core flood. This lab procedure is reproduced in a small simulation model with grid blocks comparable to the size of pore network models. These upscaling simulation models consider spatial distributions for different mixtures of rock types based on spatial correlations and transition models that are consistent with geological heterogeneity at the grid block level. The allocation of petrophysical properties estimated from pore network models is consistent with the distribution of rock types in the upscaling models. For example, the rock types in the grid may be distributed so as to represent layered systems or any other system exhibiting a spatially correlated configuration. Each upscaling model is assigned a effective rock type or a connectivity index (CI).

The results of the steady state simulation at different oil-water relative rates are used to estimate the effective petrophysical properties of the entire model, including the absolute permeability, capillary pressure and relative permeability

curves. Additionally, anisotropy analysis for these petrophysical properties can be addressed by changing the flow orientation in the model. More details about the flow-based approach to upscale flow functions is presented at the end of Chapter 6. Further developments on the issue of upscaling petrophysical properties, such as going from core flood scale to a coarse grid block in a field scale simulation, are beyond the scope of this dissertation and should be studied with more detail as an independent research topic.

3.4 PARAMETERS FOR HISTORY MATCHING

Once the macroscopic flow functions are determined based on the results of the pore network simulations and the scale up procedure, they are used as an input in the flow simulation at field scale. These flow functions represent the connection between the field scale heterogeneity models and the multiphase flow characteristics.

In the flow simulation model, an effective rock type or connectivity index (CI) is assigned to each grid block. During the history matching step, a stochastic simulation algorithm is used to model the spatial distribution of the CIs according to the expected geologic structure of the reservoir. Sequential indicator simulation is the stochastic technique used in this work; however the proposed approach can be extended to any other method for geological modeling.

The allocation of petrophysical properties in the flow simulation grid, including permeability, porosity, capillary pressure and relative permeability curves, is consistent with the distribution of the CIs. During the history matching process the distribution of connectivity indexes is perturbed in a gradual and systematic fashion to improve the match between the flow simulation response and the production history, while preserving the prior geologic model of heterogeneity. Consequently, the history matching process yields consistent representation of multiphase flow properties and the underlying geology.

The proposed approach for history matching is a modified version of the probability perturbation method for gradual deformation of geological models conditioned to dynamic information. In this approach a sequential indicator simulator is coupled with a flow simulator in a Markov-Chain application to gradually and iteratively update the reservoir model by perturbing the local conditional distributions of CIs. For this purpose, a deformation parameter that is calibrated using the production information is used. More details about the probability perturbation approach and the proposed modifications are included in the next chapter.

3.5 ASSESSING THE ACCURACY OF RESULTS

Reconciling the difference between the predictions of a simulation model and the actual production history of a reservoir is frequently a challenging task that requires intensive interdisciplinary work and a large computational effort that translates into lengthy and more expensive projects. Moreover, the history matching process is considered a complicated non-unique optimization problem characterized by, i) a significant lack of information to accurately depict the geological model; and, ii) a highly non-linear relationship between the model parameters and the simulation response. Furthermore, issues of inaccuracy in historical data and simplifying assumptions for geological modeling and flow simulation, contribute to the quality of the history-matched models in terms of their predictive accuracy.

The final purpose of every reservoir model is to provide reliable predictions of reservoir performance and explore economic alternatives to increase the recovery. Given the non-uniqueness associated with the solution of the dynamic data integration problem and the large uncertainty in the geological model and petrophysical properties, an assessment of the accuracy of the prediction model is difficult. However, it seems reasonable that models with a more accurate description of the reservoir heterogeneity in terms of petrophysical attributes would produce more realistic estimates.

Frequently, evaluation of the accuracy of a reservoir model is limited to the quantification of the match between the production and the simulated response during the period with historical records. However, given the non-unique nature of the history matching problem, a perfect match of the production history doesn't automatically imply accuracy in the future predictions of the reservoir model. This is particularly true for reservoir models calibrated with production history limited to the primary recovery information, when the production data contains rather limited information regarding the inter-well geological heterogeneity, compared to historical records that include water injection. An interesting approach for model verification for oil reservoirs with an extensive production history, is by performing the history match using the initial portion of the production history, leaving the final period of production information as a validation set. A good practice for history matched reservoir models is the continuous evaluation of the predicted performance with recently acquired production data to periodically confirm the validity of the model.

In contrast to deterministic approaches for history matching, where a single history-matched model with highly uncertain predictions is obtained, a probabilistic approach, though less common, can offer the advantage of representing the uncertainty in both model parameters and production forecast. However the probabilistic approach, generally implies history matching multiple realizations of the reservoir model that require running flow simulations over a large number of non-

matched realizations of the reservoir model. The probabilistic approach thus translates to an increased computational effort adding to the cost and time of the history matching project.

4 PROBABILISTIC HISTORY MATCHING

In this chapter, the proposed approach for history matching, based on the probability perturbation method for gradual deformation of geological models, is presented with details.

4.1 FIELD-SCALE STOCHASTIC MODELING

Reservoir models are generally constructed considering subsurface geological data obtained from different sources (such as seismic, well logging, well tests, sequence stratigraphy, etc), and a geological model of heterogeneity. The prior geological model for heterogeneity is commonly represented in the form of a variogram model that is inferred from the same conditional subsurface information. These two components are combined within a simulation/interpolation framework to generate geological models conditioned to static information.

Geostatistics as a geological modeling technique and its two original basis: i) the variogram model and ii) the kriging interpolation method, were initiated by the work of Daniel Krige (1951) and later developed by Georges Matheron (1962-1963,

1965), with the purpose of providing locally accurate grade estimates of mining blocks; however, its application has extended from the mining industry to many other related disciplines, including the oil industry.

The simple kriging (SK) estimator k_{SK}^{*} at each location $\boldsymbol{u}_j$ of the target geological model $k(\boldsymbol{u})$ (such as the permeability or porosity field) is the best linear unbiased estimator and can be written as:

$$k_{Sk}^{*} - m_k(\boldsymbol{u}_j) = \sum_{i=1}^{N} \lambda_i(\boldsymbol{u}_j)\left[k(\boldsymbol{u}_i) - m_k(\boldsymbol{u}_i)\right] \tag{4.1}$$

where, $m_k(\boldsymbol{u}_j) = E\left\{k(\boldsymbol{u}_j)\right\}$, $j = 1,...,J$, are the known stationary means of the random function $k(\boldsymbol{u}_j)$ at the locations $\boldsymbol{u}_j$; J is the size of the model; $k(\boldsymbol{u}_i), i = 1,...,N$ are the conditional data; and $\lambda_i(\boldsymbol{u}_j)$ are the kriging weights for each conditional data towards the estimation at location $\boldsymbol{u}_j$. The weights are calculated from the following system of equations:

$$\sum_{k=1}^{N} \lambda_k(\boldsymbol{u}_j) C\left(\boldsymbol{h}_{ik}\right) = C\left(\boldsymbol{h}_{ij}\right), \forall i = 1,...,N \tag{4.2}$$

where $C\left(\boldsymbol{h}_{ij}\right)$ or $C\left(\boldsymbol{u}_i - \boldsymbol{u}_j\right)$ is the covariance between the data and the unknown at spatial lag $\boldsymbol{h}_{ij} = \boldsymbol{u}_i - \boldsymbol{u}_j$, considering stationarity; and is related to the variogram by: $\gamma\left(\boldsymbol{h}_{ij}\right) = C(0) - C\left(\boldsymbol{h}_{ij}\right)$, where $C(0) = Var\left\{k\left(\boldsymbol{u}_i\right)\right\}$. The redundancy

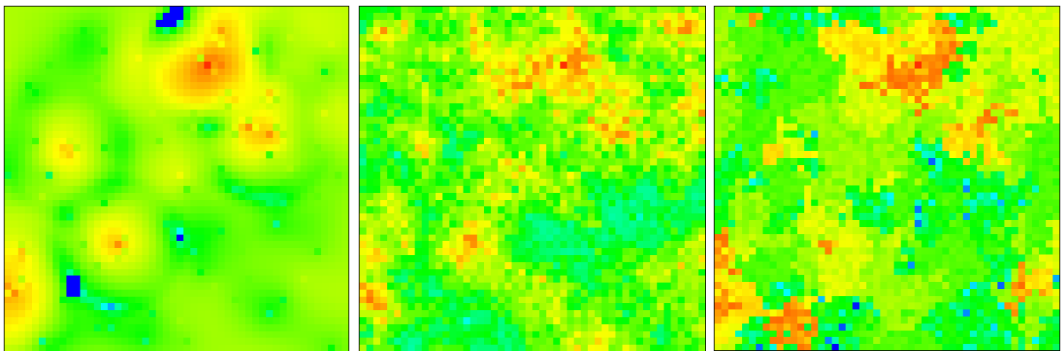
between the data is accounted through the $C(\boldsymbol{h}_{ik})$ term in the above system. The corresponding minimum estimation (error) variance σ^2_{SK} is:

$$\sigma^2_{SK}(\boldsymbol{u}_j) = C(0) - \sum_{i=1}^{N} \lambda_i(\boldsymbol{u}_j) C(\boldsymbol{h}_{ij}) \tag{4.3}$$

Stochastic simulation was introduced by Matheron (1973) and Journel (1974) to correct for the smoothing effects of kriging (see Figure 4-1), and to enable the reproduction of the spatial variance predicted by the variogram model. Different algorithms have been developed including sequential simulation (Journel, 1983, Isaaks, 1990; Srivastava, 1992; Goovaerts, 1997; Chiles and Delfiner, 1999), which has become the workhorse for many current geostatistical applications.

The stochastic simulation approaches yield the probability distributions at individual locations quantifying the uncertainty in the estimate, considering the conditional information and the spatial heterogeneity model. There are different methods for the construction of the local conditional probability distributions, including the Gaussian approach where the kriging estimation and the estimation variance are used as the mean and the variance of the local normal conditional distribution. In another approach, the use of indicator transforms allows modeling local conditional distributions without relying on Gaussian assumption. The resultant models are realizations from a multivariate, non-parametric distribution that is data-

driven and exhibit more connected and well-defined geological bodies. Figure 4-1 compares the results of the original kriging interpolation technique (a), with the result from Gaussian simulation (b) and that from Sequential Indicator Simulation (c), considering the same conditional information.



a) b) c)

Figure 4-1. Spatial interpolation obtained by: a) Kriging; b) sequential Gaussian simulation and c) sequential indicator simulation.

Stochastic simulation also provides the capability to generate multiple equiprobable realizations, giving birth to the idea of assessing spatial uncertainty (Journel and Huijbregts, 1978).

4.2 SEQUENTIAL INDICATOR SIMULATION

Spatial distributions can be modeled following a non-parametric approach, where the local conditional probability distributions $F(\boldsymbol{u}; z_i \mid (n))$ can be calculated from the available conditional information, by defining a set of thresholds

$z_i, i = 1, ..., N_T$ to discretize the range of variability of the spatial variable. The data are then transformed to binary indicator variables. Performing indicator Kriging with the indicator transformed variables can yield the corresponding non-parametric conditional distributions.

The indicator transform of a random variable is simply a binary transform: the value one is assigned if the value at a location is less than the threshold and zero if not. The indicator transform corresponding to threshold z_i is defined as:

$$I(\boldsymbol{u}, z_i) = \begin{cases} 1 & if\ k(\boldsymbol{u}) \le z_i \\ 0 & if\ k(\boldsymbol{u}) > z_i \end{cases} \quad \forall i = 1, ..., N_T \tag{4.4}$$

Considering this binary transform, the expected value of an indicator random variable is equivalent to the cumulative probability at that particular threshold. Hence, the probability distribution can be calculated by sequentially calculating the expected value of the indicator random variable corresponding to different thresholds. The expected value of the indicator random variable corresponding to a particular threshold is:

$$E\{I(\boldsymbol{u}, z_i\} = \text{Prob}\{k(\boldsymbol{u}) \le z_i\} = F_k(z_i) \tag{4.5}$$

The indicator coded data is used to infer the experimental indicator variogram at each threshold, allowing the usage of different heterogeneity models (variograms) for different thresholds. The definition of the indicator variogram is analogous to that for continuous variables:

$$\gamma_I\left(h;z_k\right)=E\left\{\left[I\left(u;z_k\right)-I\left(u+h;z_k\right)\right]^2\right\} \tag{4.6}$$

However, considering the binary nature of the indicator transformed data, only data pairs at a particular lag with values on opposite sides of the threshold contribute to this measure of spatial correlation. Consequently, the indicator variogram for a particular threshold of a random variable is a measure of variability and corresponds to the probability for the transition from one side of the threshold to the other, i.e. from one category (such as permeability range or rock type) to another.

At a particular location, the conditional expectation of the indicator random variable for each threshold is determined by applying indicator kriging with the available indicator coded conditioning data.

$$I^*\left(\boldsymbol{u}\middle|z_i;(n)\right)=F_k^*\left(\boldsymbol{u}\middle|z_i;(n)\right)=\sum_{i=1}^{n}\lambda_\alpha(\boldsymbol{u}).i(\boldsymbol{u}_\alpha;z_i) \tag{4.7}$$

where $i(\boldsymbol{u}_\alpha; z_i)$ is the indicator coded data at location $\boldsymbol{u}_\alpha$, and the weights $\lambda_\alpha(\boldsymbol{u})$ are obtained by solving an indicator kriging system that utilizes indicator covariances:

$$\sum_{\beta=1}^{n} \lambda_\beta(\boldsymbol{u}, z_i) C_I\left(\boldsymbol{h}_{\alpha\beta}; z_i\right) = C_I\left(\boldsymbol{h}_{\alpha o}; z_i\right), \forall \alpha = 1, ..., n \tag{4.8}$$

The probabilities (conditional expectations) for the local conditional distributions are evaluated at a limited set of thresholds. Therefore, interpolation and extrapolation methods are required to obtain a continuous conditional cumulative distribution function. Interpolation between the thresholds and tail extrapolations can be obtained by applying different approaches such as linear or hyperbolic interpolation/extrapolation or using tabulated values.

A realization of the target reservoir model is generated by visiting the nodes in a random order, constructing the local conditional probability distributions, randomly sampling from the local conditional distributions and then proceeding to the next location, where the previously simulated values become additional conditional information for the construction of the local conditional distribution. The process is repeated until all the uninformed locations in the model are populated. Monte Carlo or other sampling technique can be applied to sample from the local conditional distributions. Multiple realizations of the target reservoir can be obtained by altering

the random path and/or changing the random draw from the local conditional distributions.

Kriging is the driver for the above-described simulation procedure and kriging is data exact and ensures reproduction of the covariance between the data and the estimation node. Since previously simulated values are included in the set of the conditioning data, this ensures that the final simulated model reproduces the correct spatial correlation characteristics.

The application of the sequential indicator simulation approach can be summarized in the form of the following steps:

1. Select appropriate thresholds consistent with the spatial phenomena.
2. Indicator code the data corresponding to different thresholds
3. Infer indicator variogram/covariance model(s) for different thresholds.
4. Define a random path to visit all uninformed locations. At each simulation location along the random path apply the following sub-procedure:

4.1. Apply indicator kriging with the available indicator coded conditional information, to construct the conditional probability distribution corresponding to each threshold. Repeat for all thresholds.

4.2. Correct for order-relations (non-monotonicity of the distributions) of the evaluated probabilities (conditional expectations).

4.3. Randomly sample a value from the local conditional distribution. Prior to the sampling process, use interpolation/extrapolation methods to model a continuous ccdf from the discrete probabilities evaluated at the thresholds.

4.4. Include the sampled value in the list of conditional information for subsequent estimations.

5. A single realization of the target reservoir is obtained after all the uninformed locations have been visited following the random path. To generate multiple realization repeat step 4 with different random paths.

A sequential indicator simulator has been implemented in C++ language, and validated with other algorithms available in the public domain. This algorithm is the base code for the probability updating method utilized in this research. In subsequent sections, additions including modules for gradual deformation of geological models, interface with flow simulators, and optimization schemes are also developed.

4.3 DYNAMIC DATA ASSIMILATION

Honoring the geological model is an important objective during the generation of static geological models; however, it is commonly under-emphasized during the history-matching phase. In the final stages of reservoir modeling, the history matching process, the perturbations or modification in the model should be performed

to ensure a match to the flow history, while preserving the geological model of heterogeneity. In this research, that goal is accomplished by applying a probabilistic approach for gradual deformation of geological models. The gradual deformation is achieved by systematically perturbing the local conditional distributions with a deformation parameter, r_D that is calibrated using the available dynamic information.

4.3.1 MERGING INFORMATION FROM DYNAMIC AND STATIC SOURCES

The local conditional probability distributions obtained during sequential indicator simulation reflect the constraining information from conditioning "hard" data and the prior geological (semi-variogram) model. In many development scenarios, the requirement is to update these geologically consistent reservoir models using production information gathered at a few development wells. This requires merging the local conditional distribution P(A|B), obtained by indicator kriging, with additional information P(A|C), that is derived on the basis of the dynamic (production) information. Here A represents the geological attribute at a certain location that is the objective of the simulation; B represents the "hard" conditioning data and the prior geological information; and C is the dynamic or production information. We start of with a discussion on the method for merging conditional

information measures derived from multiple sources. Subsequently, we detail the procedure for calibrating the information from dynamic data i.e. P(A|C).

The Permanence of Ratio Hypothesis (Journel, 2002) is the method applied to combine the individual distributions conditioned to dynamic and static information. The following distance or information measures *a*, *b*, *c* and *x* are defined:

$$a = \frac{1-P(A)}{P(A)} \quad b = \frac{1-P(A|B)}{P(A|B)} \quad c = \frac{1-P(A|C)}{P(A|C)} \quad x = \frac{1-P(A|B,C)}{P(A|B,C)} \qquad (4.9)$$

The quantity *a*, for example, denotes the relative distance to the event A based on the prior probability P(A). If P(A) is one, the relative distance a is zero. The relative distance a is infinity, if P(A) is zero. The measures *b*, *c* and *x* can be interpreted similarly, on the basis of the conditional probabilities P(A|B), P(A|C) and P(A|B,C).

According to the permanence of ratios hypothesis, the relative updating of a simulation event (A) due to a dynamic event (C) remains the same irrespective of the presence of the static event (B). This can be written in terms of *a*, *b*, *c* and *x* as:

$$\frac{x}{b} = \frac{c}{a} \Leftrightarrow x = \frac{bc}{a} \qquad (4.10)$$

Consequently, the joint probability of the simulation event given the dynamic and static information can be calculated from the elemental probabilities – the prior probability for A , P(A); the conditional probability of A given the dynamic information, P(A|C); and the conditional probability of A given the static information B, P(A|B):

$$P(A \mid B,C) = \frac{1}{1+x} = \frac{a}{a+bc} \tag{4.11}$$

In this approach, the marginal probabilities of the static data P(B), and that of the dynamic data P(C), which are difficult to estimate in practice, are not required.

The Permanence of ratio hypothesis was derived from the full and conditional independence hypothesis – popular Bayesian methods for data integration and statistical inference, resulting in a more robust and consistent approach to estimate joint probabilities. However, as its predecessors, the permanence of ratio hypothesis still relies in simplifying assumptions that disregard the strict data interdependence. A tau model for introducing such dependence has been proposed by Journel (2002) and Krishnan (2004), however a practical method for calibrating the interdependency parameter tau, remains elusive.

The permanence of ratio hypothesis can be generalized to introduce other data events. Consider, for example, the probability of the geological attribute A,

conditioned to seismic information, D – P(A|D). The following relative distances are introduced

$$d = \frac{1 - P(A \mid D)}{P(A \mid D)} \quad x' = \frac{1 - P(A \mid B, C, D)}{P(A \mid B, C, D)} \tag{4.12}$$

The relative contribution of D to the event A is independent of the presence of B and C. This can be expressed as

$$\frac{x'}{x} = \frac{d}{a} \Leftrightarrow x' = \frac{xd}{a} = \frac{bcd}{a^2} \tag{4.13}$$

And finally the probability of the geological even A, conditioned to B, C and D can be expressed under the permanence of ration hypothesis as:

$$P(A \mid B, C, D) = \frac{1}{1 + x'} = \frac{a^2}{a^2 + bcd} \tag{4.14}$$

A primary requirement in the implementation of the above scheme, is the availability of the local probability distributions of the geological event A, (rock type, permeability, porosity, etc) conditioned to the dynamic information, C, i.e. P(A|C). The probability perturbation approach for gradual deformation of reservoir models can be used to calibrate the local probability distributions of the spatial geological attributes conditioned to dynamic data, i.e. P(A|C). This approach requires the implementation of an optimization scheme to calibrate a deformation parameter, and the development of interfaces between the geological modeling algorithm and a flow

simulator. Details about the implementation of the probability perturbation approach are presented in the next section.

4.3.2 GRADUAL CALIBRATION OF RESERVOIR MODELS

Gradual deformation approach relies in modeling a systematic and smooth transition between realizations of a stochastic model while preserving their spatial variability. In this approach, the unknown parameter space (the entire population of a reservoir attribute in a finite difference model) in the history-matching problem is significantly reduced to one (or few) parameter(s). Thus, this method is considered a reduced parameterization approach and can be used in an effective iterative optimization procedure for dynamic data integration. Additionally, the gradual deformation method honors the statistical properties of the spatial geological attribute. A particular approach for gradual deformation is the probability perturbation method where the underlying local probability distributions, instead of the actual properties, are perturbed to model a smooth transition between stochastic realizations.

The probability perturbation method for gradual deformation starts with a particular realization of the target reservoir, such that all locations in the reservoir, cells or nodes have attribute values. In the case of continuous variables, the initial

realization is transformed into an indicator random field such that the value at a particular location falls within an indicator category referred to as the initial class, $I_k, k = 1, \ldots, K+1$.

4.3.2.1 Original Perturbation Scheme

During the gradual deformation of the geological model, the local conditional distributions are perturbed over a ℓ step iterative process using a dimensionless deformation parameter r_D between [0,1]. The updated category $I_{k'}$ at each location is computed from the perturbed distribution. The parameter r_D thus controls the transition of category I_k at a location $\boldsymbol{u}$ to a new category $I_{k'}$. The original probability perturbation method considers the following iterative scheme to update the local conditional probability distributions from step ℓ to step $\ell+1$ considering the parameter r_D, calibrated with the dynamic information (denoted by C).

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_{k'} \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = P\left(A^{\ell+1} = I_{k'} \mid C\right) = r_D \cdot P\left\{I(\boldsymbol{u}) = I_{k'}\right\}, \quad \forall k' \neq k$$

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_k \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = P\left(A^{\ell+1} = I_k \mid C\right) = 1 - \sum_{k' \neq k} r_D \cdot P\left\{I(\boldsymbol{u}) = I_{k'}\right\} \quad (4.15)$$

The probability $P\left\{I(\boldsymbol{u}) = I_{k'}\right\}$, denoted also as P(A), is the prior marginal probability corresponding to indicator category $I_{k'}$, calculated from the available

data. The optimum value of the deformation parameter r_D is calibrated so as to reduce the mismatch between the simulated response and the production history. Upon convergence, the updated left hand side conditional probability $P\left\{I^{\ell+1}(\boldsymbol{u}) \mid I^{\ell}(\boldsymbol{u}), C\right\}$ becomes P(A|C), i.e. the conditional probability that quantifies the information in the dynamic data regarding the spatial variations of the geological attribute in the reservoir. The probability for the transition from category I_k to a new category $I_{k'}$ increases with the value of r_D. Therefore, the parameter r_D controls the gradual transition between the initial realization of the geological attribute (recovered with $r_D = 0$) and a new equiprobable stochastic realization (for $r_D = 1$). Consequently, lower values of r_D generate realizations similar to the initial realization, while higher values may result in significantly different realizations.

The procedure requires the generation of multiple equiprobable stochastic realizations of the geological attribute, including the initial realization. These stochastic realizations are generated using sequential indicator simulation with the available static data and a structural model for geological heterogeneity, denoted as B. To minimize the possibility of getting stuck in a local minimum during the search for a global optimum for r_D, the random path as well as the random draws from local conditional distributions are modified during the generation of the new proposed stochastic realizations.

The limiting cases of the perturbation scheme in the probability perturbation method can be confirmed under the permanence of ratio hypothesis.

For $r_D = 0$:

Initial Class I_k	Other Classes $I_{k'}$
$P(A = I_k \mid C) = 1 - \sum_{k' \neq k} r_D \cdot P(A = I_{k'}) = 1$	$P(A = I_{k'} \mid C) = r_D \cdot P(A = I_{k'}) = 0$
$c = \frac{1 - P(A = I_k \mid C)}{P(A = I_k \mid C)} = 0$	$c = \frac{1 - P(A = I_{k'} \mid C)}{P(A = I_{k'} \mid C)} = \infty$
$P(A = I_k \mid B, C) = \frac{a}{a + bc} = 1$	$P(A = I_{k'} \mid B, C) = \frac{a}{a + bc} = 0$

The merged probability of the initial class is one, while the probability of the other classes is zero. Consequently the initial realization is recovered.

For $r_D = 1$:

Initial Class I_k	Other Classes $I_{k'}$
$P(A = I_k \mid C) = 1 - \sum_{k' \neq k} r_D \cdot P(A = I_{k'}) = P(A = I_k)$	$P(A = I_{k'} \mid C) = r_D \cdot P(A = I_{k'}) = P(A = I_{k'})$

$$c = \frac{1 - P(A \mid C)}{P(A \mid C)} = \frac{1 - P(A)}{P(A)} = a$$

$$P(A \mid B, C) = \frac{a}{a + bc} = \frac{1}{1 + b} = P(A \mid B)$$

Thus, for $r_D = 1$ the proposed realization condition to the hard data and the geological information, P(A|B) is recovered. Since the random path and the random draws are changed, this amounts to a new realization sampled from P(A|B).

Figure 4-2 shows an example of gradual deformation of geological models by probability perturbation method under the first scheme. The single parameter, r_D, determines the magnitude of the perturbation in this model. As is evident, the perturbation transforms an initial model ($r_D = 0$) to a completely new model ($r_D = 1$) through a succession of steps with small variations.

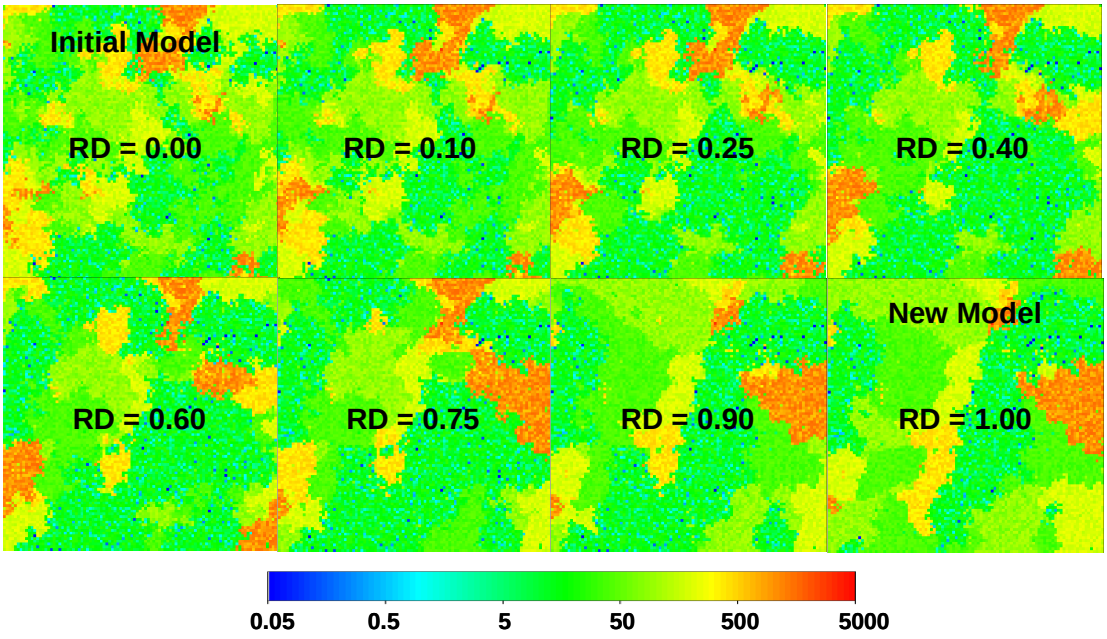


Figure 4-2. Example of gradual deformation of geological models by probability perturbation method.

In this original scheme the deformation parameter, r_D reduces the probabilities of all indicator categories in the local conditional distribution, except that of the initial class, I_k, which is proportionally increased. This is because in this case the deformation parameter, $r_D \in [0,1]$, multiplies the probabilities of the other indicator categories, ($k' \neq k$) and the probability of the initial class always increases (or remains the same for r_D = 1). Since the deformation parameter only increases the probability of the initial class, the maximum deformation of the model is rather small, slowing the process of gradual deformation.

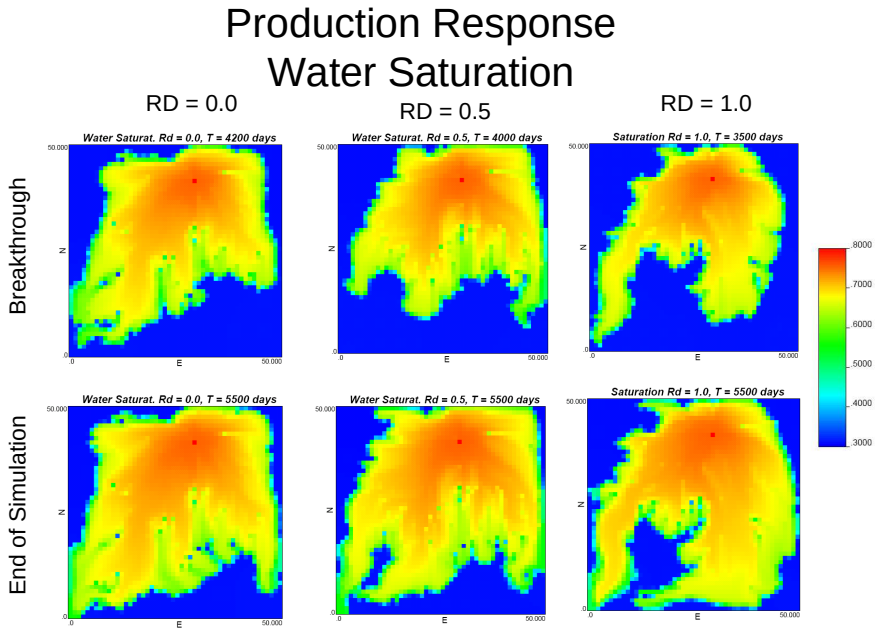


Figure 4-3. Effect of the r_D parameter in the water saturation distribution in a reservoir model at breakthrough (~4000 days) and the end of simulation (5200 days).

Figure 4-3 and Figure 4-4 show the impact of the deformation parameter on the flow response during an application of the probability perturbation method for the gradual deformation of the geological model in Figure 4-2.

Different alternate perturbation schemes for the local conditional probability distributions, additional to the original presented above, have been evaluated to define the method that better fits the objectives of the research and improves the convergence characteristics in the search of an optimal r_D. These alternative perturbation schemes are presented in the next sections.

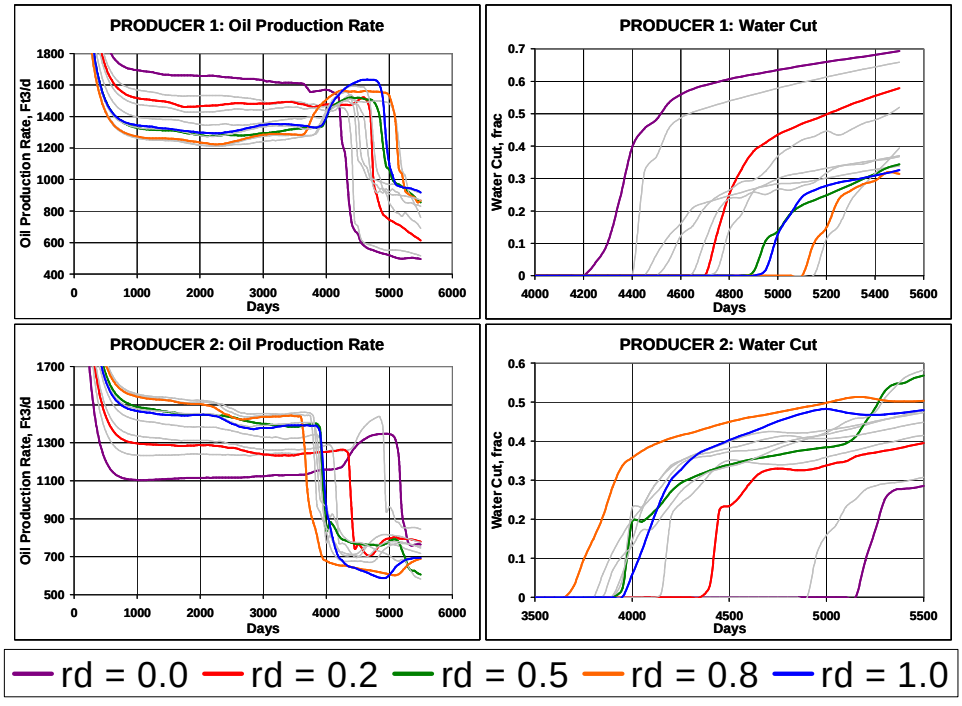


Figure 4-4: Effect of the deformation parameter on the flow response of a geological model.

4.3.2.2 Second perturbation scheme

In this second perturbation scheme, the deformation parameter reduces the probability of the initial class in the local distributions, while the probabilities of the other indicator categories increase proportionally. In this case the probability of the initial class is always reduced (or remains the same for r_D = 1). This perturbation scheme can be written as:

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_k \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = r_D \cdot P\left\{I(\boldsymbol{u}) = I_k\right\} \tag{4.17}$$

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_{k'} \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = \frac{P\left\{I(\boldsymbol{u}) = I_{k'}\right\}\left(1 - r_D \cdot P\left\{I(\boldsymbol{u}) = I_k\right\}\right)}{\sum_{k' \neq k} P\left\{I(\boldsymbol{u}) = I_{k'}\right\}}, \quad \forall k' \neq k$$

The perturbed conditional probability for $k' \neq k$ is written such that the perturbed probability for a particular class is scaled according to its initial value. A deformation parameter value of zero will generate a distribution with probability 0 for the initial class, producing a notable deformation of the geological model. A deformation parameter of value one will recover a new proposed realization conditioned to geological information. Now, the deformation parameter only decreases the probability of the initial class, ensuring a large deformation of the geological model, speeding the process of gradual deformation. However, this perturbation scheme does not allow the reproduction of the initial realization and consequently the deformation process is no longer systematic and controlled.

4.3.2.3 Third perturbation scheme

In the third perturbation scheme, the two previous schemes were combined to ensure a more controlled, but at the same time fast gradual deformation of the geological model. In this case the probability of the initial class has a range of variation from 1 to 0 (for r_D = 0 to r_D = 1), ensuring the reproduction of the initial realization for r_D = 0.0, the recovery of the new distribution conditioned to geological information for r_D = 0.5, and eliminating the probability of the initial class in this new conditional distribution for r_D = 1.0. In this approach the range of variation of the deformation parameter, r_D, is divided in two intervals, values below and above 0.5. For values of r_D below 0.5, the original perturbation scheme is applied using the transformation $r_D^{'} = 2r_D$. For r_d values above 0.5, the second perturbation scheme is applied with the transformed value $r_D^{'} = (2 - 2r_D)$.

The rational for splitting the range of r_D is that the combined application of the two first perturbation schemes for an increased range of variation in the local conditional distribution ranging from the close reproduction of the initial model for small values of r_D, such as $r_D < 0.2$, to the rejection of this initial realization for $r_D > 0.8$. The gradual transition between these two extremes is modeled through a

new realization, which is recovered at $r_D = 0.5$. The perturbation scheme can be written as:

$$r_D^{'} = \begin{cases} 2r_D, & r_D \le 0.5 \\ 2 - 2r_D, & r_D > 0.5 \end{cases}$$

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_k \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = \begin{cases} 1 - \sum_{k' \neq k} r_D^{'} \cdot P\left\{I(\boldsymbol{u}) = I_{k'}\right\}, & r_D \le 0.5 \\ r_D^{'} \cdot P\left\{I(\boldsymbol{u}) = I_k\right\}, & r_D > 0.5 \end{cases} \tag{4.18}$$

$$P\left\{I^{\ell+1}(\boldsymbol{u}) = I_{k'} \mid I^{\ell}(\boldsymbol{u}) = I_k, C\right\} = \begin{cases} r_D^{'} \cdot P\left\{I(\boldsymbol{u}) = I_{k'}\right\}, & \forall k' \neq k, r_D \le 0.5 \\ \dfrac{P\left\{I(\boldsymbol{u}) = I_{k'}\right\}\left(1 - r_D^{'} \cdot P\left\{I(\boldsymbol{u}) = I_k\right\}\right)}{\sum_{k' \neq k} P\left\{I(\boldsymbol{u}) = I_{k'}\right\}}, & \forall k' \neq k, r_D > 0.5 \end{cases}$$

Consequently, the probability of the initial class increases for r_d values below 0.5; and decreases for rd values above 0.5, ensuring a wider but controlled range of variation in the perturbation of the local distribution. Figure 4-5 shows the effect of the deformation parameter r_D on the merged local conditional distribution, P(A|B,C), under the perturbation scheme 3. For r_D = 0.5, the local probability distribution (green curve) corresponding to the new proposed realization conditioned to geological information is recovered. The local conditional distribution for $r_D = 1$ corresponds to the new local distribution ($r_D = 0.5$), after eliminating the probability of the initial class (compare green and blue curves). In the figure, the initial class is represented by the shaded area.

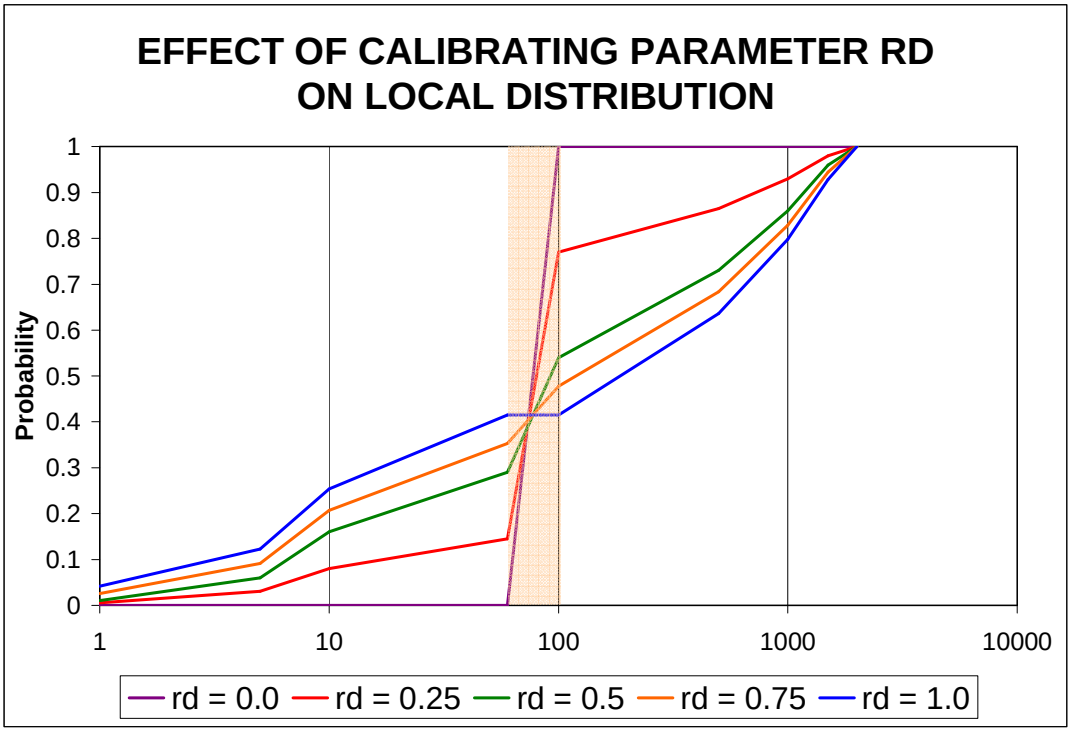


Figure 4-5. Effect of deformation parameter on local distribution with the 3rd perturbation scheme.

4.3.2.4 Final perturbation scheme

The above three perturbation schemes focus on the gradual transition between an initial realization and a new realization or proposal. Multiple equiprobable proposals can be generated using sequential indicator simulation, simply by modifying the random path and/or the set of random drawings for the local conditional distributions. However, in the three previous schemes, starting from an initial realization, the transition to only one new realization is evaluated at a time.

This implies that only the range of variability between these two limiting realizations is searched for an optimal value of r_D. This potentially can limit the convergence rate of the proposed perturbation schemes. A fourth perturbation scheme is proposed to circumvent this drawback.

The fourth perturbation scheme allows the gradual transition from an initial realization to a number of new realizations or proposals. Compared to the original gradual deformation method (Kashib and Srinivasan, 2006) where only a single new realization is evaluated at the time, the implemented scheme allows the simultaneous evaluation of four different proposals improving the range of variation in the search for an optimal value of r_D during the gradual deformation of the geological model. The transition between the initial realization and each of the four proposals takes place using the first perturbation scheme with a transformed value of r_D, $r_D^{'} = f\left(r_D\right)$. The final perturbation scheme can be written as:

$$r_D^{'} = \begin{cases} 1-4r_D, & r_D \le 0.25 \\ 4r_D - 1, & 0.25 < r_D \le 0.5 \\ 3-4r_D, & 0.5 < r_D < 0.75 \\ 4r_D - 3, & r_D \ge 0.75 \end{cases}$$

$$P\left\{I^{\ell+1}\left(\boldsymbol{u}\right) = I_{k'} \mid I^{\ell}\left(\boldsymbol{u}\right) = I_k, C\right\} = r_D^{'} \cdot P\left\{I\left(\boldsymbol{u}\right) = I_{k'}\right\}, \quad \forall k' \neq k$$

$$P\left\{I^{\ell+1}\left(\boldsymbol{u}\right) = I_k \mid I^{\ell}\left(\boldsymbol{u}\right) = I_k, C\right\} = 1 - \sum_{k' \neq k} r_D^{'} \cdot P\left\{I\left(\boldsymbol{u}\right) = I_{k'}\right\} \tag{4.19}$$

Where $P\left\{I^{\ell+1}(\boldsymbol{u}) \mid I^{\ell}(\boldsymbol{u}), C\right\}$ is the updated local probability distribution conditioned to dynamic information. The perturbation introduced by the deformation parameter is controlled to generate a gradual transition between the initial realization and four different equiprobable realizations obtained with different random paths and different random draws. In this scheme the initial realization is reproduced for r_D values of 0.25 and 0.75 (probability of the initial class is 1). The distributions conditioned to geological information corresponding to the four proposals are recovered for r_D values of 0.0, 0.5-, 0.5+ and 1.0. Consequently, in this approach the range of variation of the deformation parameter, r_D, is divided in four intervals corresponding to the ranges [0.0 – 0.25], [0.25 – 0.5], [0.5 – 0.75] and [0.75 – 1.0]. Different random paths and draws are used for the generation of the new proposals corresponding to each of these intervals. The transition between the initial realization and each of the four proposals corresponding to the intervals above thus takes place using the same perturbation scheme, but using different transformations of the deformation parameter r_D, $r_D^{'} = f\left(r_D\right)$.

The fourth scheme compared to the first three, evaluates four times the number of proposals, increasing the rate of convergence and reduces the probability of getting trapped on local minima. The following figure shows an example of gradual deformation of geological models by probability perturbation under the fourth

scheme. A single parameter, r_D determines the transition between an initial realization and four proposals.

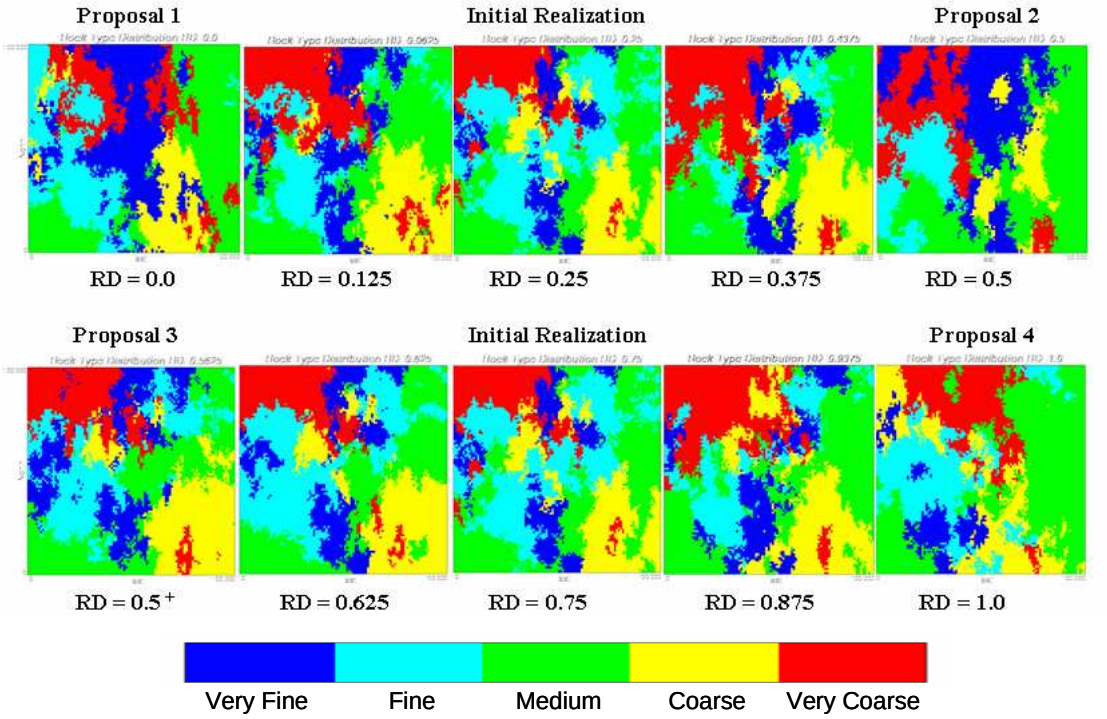


Figure 4-6. Illustration of the gradual transition between model realizations using the probability perturbation approach.

4.3.3 DEFORMATION PARAMETER OPTIMIZATION

During the history matching process, the perturbations or modification in the model should be performed to ensure a match to the flow history. Now that the method for the perturbation of local conditional distributions that form the basis for

geological models has been defined, the next step is to calibrate the deformation parameter on the basis of the observed production history of the reservoir.

Since the perturbation of the entire reservoir model is achieved using a single r_D parameter, and from Figure 4-4, it is evident that the variation of the reservoir flow response with r_D is not strictly monotonic, a simple, non-gradient based optimization method is required for the calibration of r_D. In this research, the deformation parameter r_D is calibrated using the Dekker-Brent iterative optimization procedure where the objective is to improve the fit between the flow response of the model (from the simulator) and the production history. The Dekker-Brent algorithm is an inverse parabolic interpolation method that has the advantage of being a non-gradient based approach that only requires the calculation of the objective function corresponding to different values of the deformation parameter. The algorithm yields an optimal value of the deformation parameter, r_D^* (abscissa), corresponding to a minimum value of the objective function, $\Delta O = f(r_D^*)$ (ordinate). Three abscissa values are required, *a*, *b* and *c* with the corresponding values of the objective function *f(a)*, *f(b)* and *f(c)*; and *b* chosen such that $a < b < c$ and $f(a) > f(b) < f(c)$. The estimated location of the abscissa (r_D^*) with the apparent minimum ordinate (objective function) is calculated by fitting a parabola through these three points.

$$r_D^* = \frac{\frac{(b+c)\,f(a)}{(a-b)(a-c)} + \frac{(a+c)\,f(b)}{(b-a)(b-c)} + \frac{(a+b)\,f(c)}{(c-a)(c-b)}}{2\left(\frac{f(a)}{(a-b)(a-c)} + \frac{f(b)}{(b-a)(b-c)} + \frac{f(c)}{(c-a)(c-b)}\right)} \tag{4.20}$$

The next figure illustrates the process of convergence of an objective function for a single-parameter problem using the Dekker-Brent inverse parabolic interpolation. An inverse parabola (doted red) is fitted through the initial values of r_D, *a*, *b* and *c* and the correspondent objective functions for these values, *f(a)*, *f(b)* and *f(c)*, resulting in the first apparent optimal value of r_D = *x1*. Then, the real objective function correspondent to *x1*, *f(x1)* is calculated and based on the result, the three points for the next parabolic interpolation (doted green) are selected (points *a*, *b* and *x1*). Each iteration, the set of three points is updated with the one that has the minimum calculated objective function and the two adjacent points, one on each side. This iterative procedure continues until a minimum in the objective function is reached.

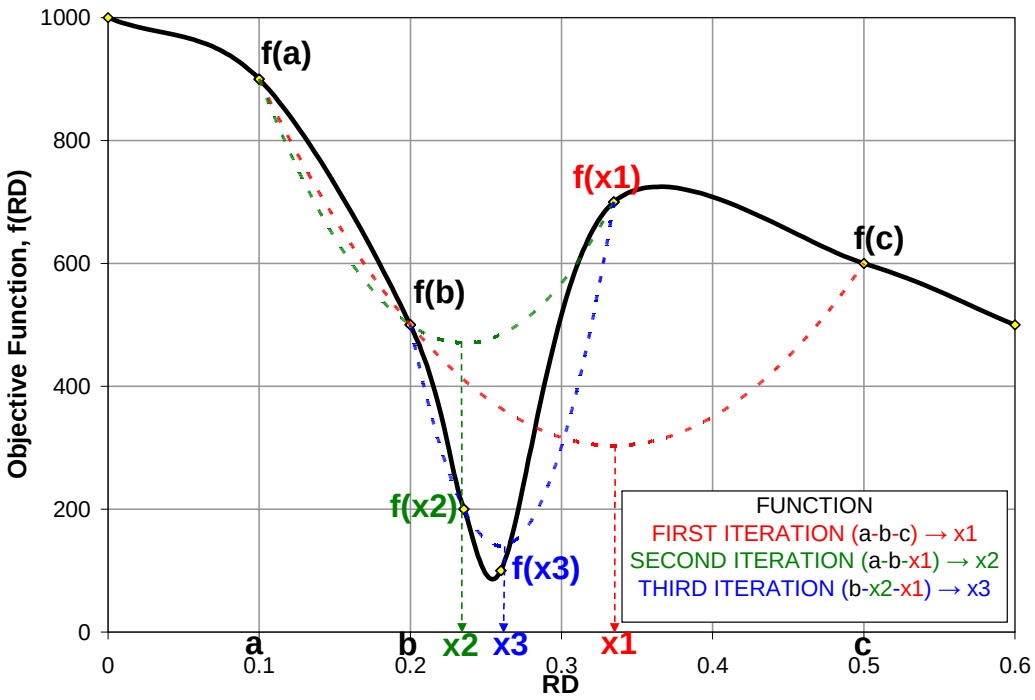


Figure 4-7. Illustration of the Dekker-Brent inverse parabolic interpolation process for determining the optimal r_D.

4.3.4 OBJECTIVE FUNCTION

In this case, the objective function to be minimized is a measurement of the deviation between the simulated production response and the production history. Different production variables can be included in the objective function, including field and well pressures, single phase rates, ratios of phase flow rates, and basically any other variable that can be calculated by the flow simulator and for which a historical record is available. The mismatches of individual variables are normalized to regulate their influence on the objective function. However, in some cases it might

be useful to assign higher weights to some production variables to emphasize the relevance of their reproduction in the target model. The proposed L^2 Norm objective function for N production variables over T time steps is:

$$Obj\ Fun = \sum_{t=1}^{T} \sum_{i=1}^{N} \left[\frac{\left(Sim_{i,t} - Hist_{i,t} \right)^2}{Var\left(Hist_i \right)} \right] \tag{4.21}$$

Where $Sim_{i,t}$ represents the simulated value of the production variable i at time t, and $Hist_{i,t}$ represents the correspondent historical value for the same variable at the same time. The square of residuals between the simulated and the historical values at a particular time is normalized by the variance of the historical values over time, to control the influence of each variable on the objective function. Other objective functions can be easily implemented applying different norms and normalization methods.

4.4 GRADUAL UPDATING PROCEDURE FOR HISTORY MATCHING

At this point, the method to estimate the local distribution conditioned to dynamic information, P(A|C), has been presented, i.e. perturbing the local conditional distributions with a deformation parameter r_D calibrated with the production history data. To ensure consistency with the geological model during all stages of the history matching procedure, the permanence of ratio hypothesis is applied to merge the conditional distribution P(A|C) with the distribution inferred from “hard” data and

prior geological model, P(A|B). Realizations sampled from the resultant merged distribution P(A|B,C) will honor both the static information (well data and geologic interpretation) as well as the historic production data.

The reproduction of historical production data is a complex non-linear inverse problem. This implies that the optimization process cannot be accomplished in a single perturbation loop starting from an initial realization reservoir model; calibrating an optimal r_D value and obtaining an updated probability distribution reflecting the dynamic characteristics of the reservoir. Instead, a multi-loop iterative process is required to update the geological model using the dynamic data.

A Markov-Chain is a stochastic updating procedure where the parameter state at any step of the procedure is assumed to be dependent only on the state immediately prior to that step. Thus the proposed realization at any stage of the process depends only on the preceding realization in the sequence, and the convergence towards the desired target depends on carefully specifying the transition from one realization to the next one, i.e. the method for the new proposed realization. In this case, the parameter r_D controls the transition of the geological parameter value at a location from one category to the next. The converged parameter state can be proved to be independent of the starting guess, provided the mechanism for transition is licit and follows some closure properties. In the current context, this implies that the

implementation of the Markov-Chain will yield history-matched models that are insensitive to the starting guess.

In the implemented Markov-Chain approach, at every updating step, from iteration step ℓ to step $\ell+1$ of an outer loop, the probability distributions conditioned to dynamic and static information, $P(A|B,C)^{\ell}$, is obtained by applying the permanence of ratio hypothesis to combine distributions conditioned to static and dynamic information. The distribution $P(A|B)^{\ell}$ is obtained from geological data and heterogeneity model. The distribution conditioned to dynamic information, $P(A|C)^{\ell}$ is estimated iteratively knowing the indicator category at each location from the realization sampled from $P(A|B,C)^{\ell}$, the prior distribution P(A) and the deformation parameter, r_D, calibrated using the Dekker-Brent iterative optimization procedure. At the end of each inner Dekker-Brent loop the converged distribution $P(A|B,C)^{\ell}$ is used to sample the initial realization for the next outer loop, $\ell+1$. The procedure is continued until global match to the historic data is attained. The calibration of the deformation parameter r_D to honor the available dynamic information thus represents the internal optimization scheme. The converged model and realization at the end of the inner loop is used as the starting realization for the next sequence of inner Dekker-Brent optimization runs to determine the conditional distribution P(A|C) and that constitutes the outer optimization loop.

Even though the two-loop Markov chain procedure ensures global convergence, the introduction of multiple sets of inner optimization schemes requires multiple evaluations of the flow response. However, the gradual deformation method renders the history match process faster and more controlled, increasing the consistency between the initial and the proposed realizations at every step, and improving the rate of convergence of the objective function.

4.5 IMPLEMENTATION OF THE METHOD

The calibration of the dynamic parameter r_D and the subsequent merging of conditional probability distributions require the implementation of an interface between the geological modeling algorithm and the flow simulator. In this project, the geological modeling algorithm (Sequential Indicator Simulation) is the main program that has been expanded to include all the tasks in the probabilistic approach for dynamic data integration. The flow simulator (ECLIPSE® from Schlumberger) is also executed within the program.

The task of combining the conditional probability distributions P(A|B) and P(A|C) into a joint-conditional distribution P(A|B,C) to sample a new realization of the geological model is also implemented within the main program (see code in

Appendix C). The geological model used in the flow simulator to calibrate the deformation parameter r_D is sampled from this jointly conditioned distribution, P(A|B,C). The complete probability updating procedure therefore consists of:

1. A sequential indicator simulator is used to generate an initial stochastic realization of the target reservoir model and calculate the local probability distributions conditioned to static information.
2. A Markov-Chain iterative updating process is started with the initial realization. The Markov chain forms the outer loop of the procedure and every outer step or outer iteration includes the following sub procedures:

2.1. Generate new random paths and sets of random draws to sample from the local conditional distributions. The random paths and the sampling draws are fixed during each outer iteration, but changes from one outer iteration to the next.

2.2. Evaluate the Objective function at different values of the deformation parameter, r_D, spanning the whole range of variability [0, 1]. Usually 9 different values of r_D are used to evaluate four new different proposals and the transition from the initial realization. For each value of r_D, a different geological model (rock type or permeability) is generated and run in the flow simulator (ECLIPSE®) to obtain the production response and the corresponding objective function.

2.3. Pick the value of the deformation parameter with the minimum objective function and start the calibration process of r_D with the dynamic data using the Dekker-Brent iterative algorithm. This calibration process is called the inner loop, and the number of inner steps or inner iterations can be fixed or controlled by a tolerance in the change of the objective function in consecutive steps.

2.4. Use the best model (with the minimum objective function) to update the stochastic realization. When the best model is obtained with a deformation parameter of 0.25 or 0.75, no updating is required (the realization remains invariant). On the other hand, if the lowest objective function value corresponds to r_D equal 0, 0.5^-, 0.5+ or 1; one of the 4 alternate proposals is selected and the outer Markov-Chain process is continued.

3. Repeat step 2 (outer loop) until a tolerance in the objective function (history match) has been reached or for a fix number of outer iterations.

4. Print out final rock type or permeability realization with the corresponding flow response.

Summarizing, the algorithm couples a modified sequential indicator simulator with a flow simulator in a Markov-Chain approach where the reservoir model is gradually updated in an iterative process. The local conditional distributions of rock type are updated using a deformation parameter that is calibrated using the production

information. This dynamic parameter, which is calibrated using the Dekker-Brent iterative optimization procedure, determines the magnitude of perturbation of the local distribution. Local distributions conditioned to static and dynamic information are iteratively updated and sampled until a global match to the historic data is attained. The algorithm and examples of input files are presented in Appendices C and D, respectively.

4.6 PRELIMINARY VALIDATION

The probability perturbation method for gradual deformation of geological models conditioned to dynamic information is implemented on a synthetic case study. This approach, compared to other perturbation methods, offers the important advantages of preserving the prior geological heterogeneity model and simplifying the history match process to a single (or few) parameter(s) optimization problem.

The details of the synthetic case are described in Table 4-1. For reference, the "truth" model used to generate the synthetic history is shown in Figure 4-8. The gradual perturbation of the reservoir model is shown in Figure 4-9. Although the complete model is in 3-D, the update process in Figure 4-9 is shown only for the third layer. Results after 45 flow simulation runs, comprising of 5 outer iterations with 9 inner Dekker-Brent iterations, are presented. Figure 4-10 and Figure 4-11 show the

field pressure and production history match obtained with the probabilistic dynamic data integration algorithm for the simulation case. Figure 4-12 shows the convergence of the objective function in the study case. During the iterative process, the relatively gradual changes to the objective function are crucial for preventing the procedure from getting trapped in a local minimum.

SIMULATION PROPERTY/DESCRIPTION	VALUE
Simulation Model	Black Oil
Solution	Implicit
Simulation Period, years	2
Grid (Cartesian)	50x50x5
Active Grid blocks	12500
Grid block dimensions, ft^3	80x80x4
Porosity	0.22
Kx = Ky (Mean – Std Dev), md	200 - 250
Kz/Kx	0.15
Saturation Pressure, psi	5064
Water-Oil Contact, ft	9000
Reference Depth, ft	7300
Initial Pressure @ 7300 ft, psi	6000
Residual Water Saturation	0.18
Residual Oil Saturation	0.24
Oil Relative Permeability Endpoint	0.7
Water Relative Permeability Endpoint	0.5
Oil Gravity (API)	35
Water Injectors	1
Injection – Control Rate, Stb/day	5000
Injection – BHP upper Limit, psi	8000
Oil Producers	2
Production – Control BHP Lower limit, psi	2000
Production – Minimum rate, Stb/day	10

Table 4-1. Description of a simulation case for the preliminary evaluation of the history matching algorithm.

The history matched model clearly exhibits the preferential connectivity between the injector and the first producer (bottom left) through high permeability paths. It also identifies the low permeability area that surrounds the second producer (bottom right). Even though the proportion of high permeability zones in the third layer appears to be higher in the history match model compared to the reference model; this proportion is similar in both models when all the layers are considered. The reference model was built with increasing proportions of high permeability regions towards the bottom of the model, while the models used during the history matching process consider a continuous heterogeneity with depth.

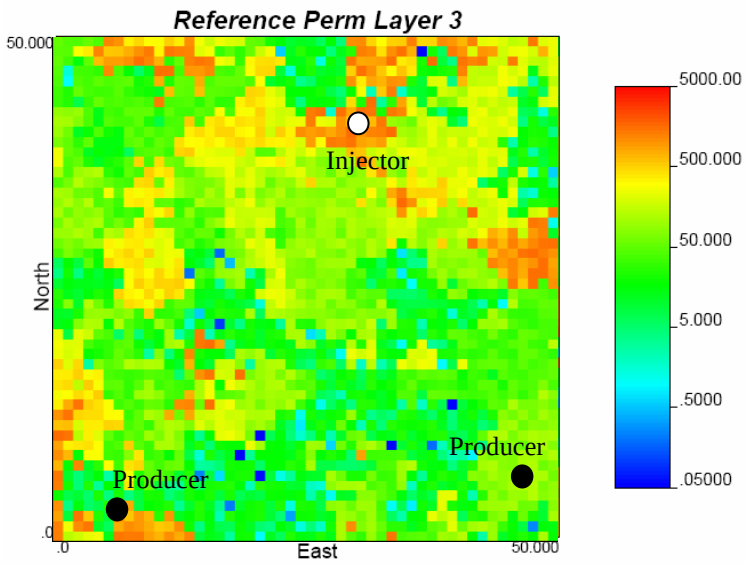


Figure 4-8. Third layer of the reference geological model in the validation case.

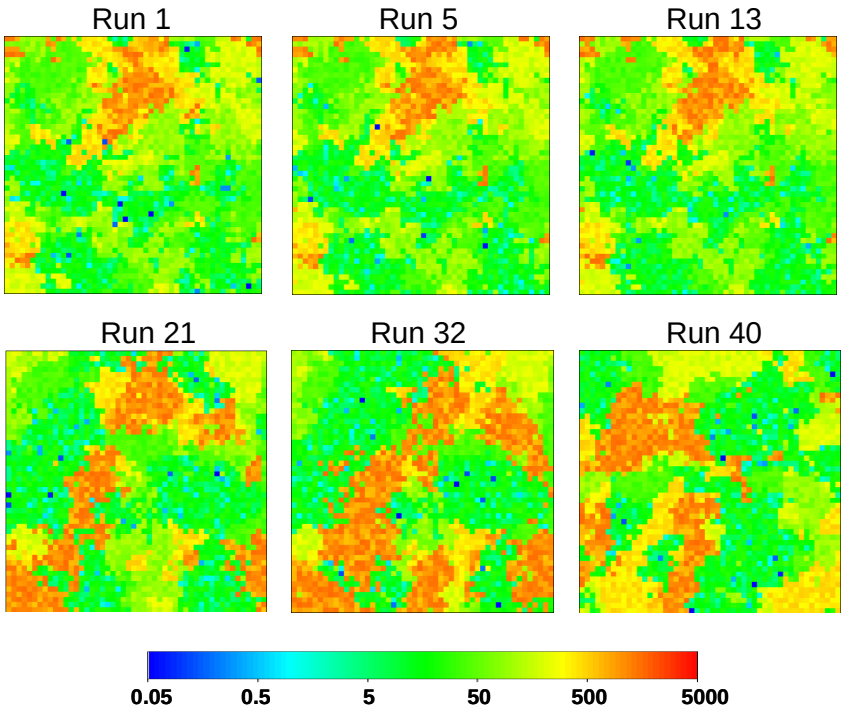


Figure 4-9. Gradual deformation of the geological model (third layer) in the validation case.

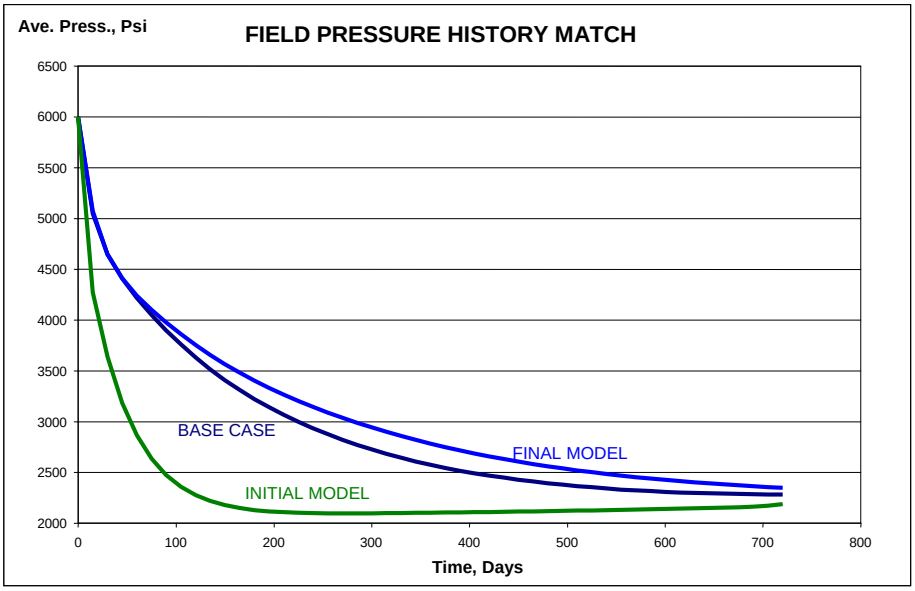


Figure 4-10. Field pressure history match for the validation case.

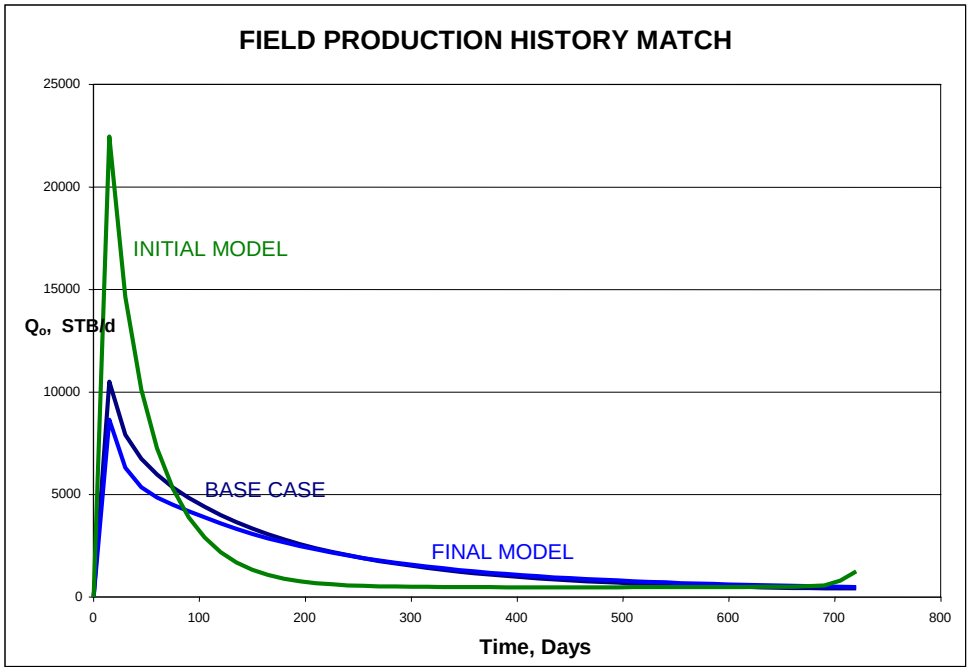


Figure 4-11. Field production history match for the validation case

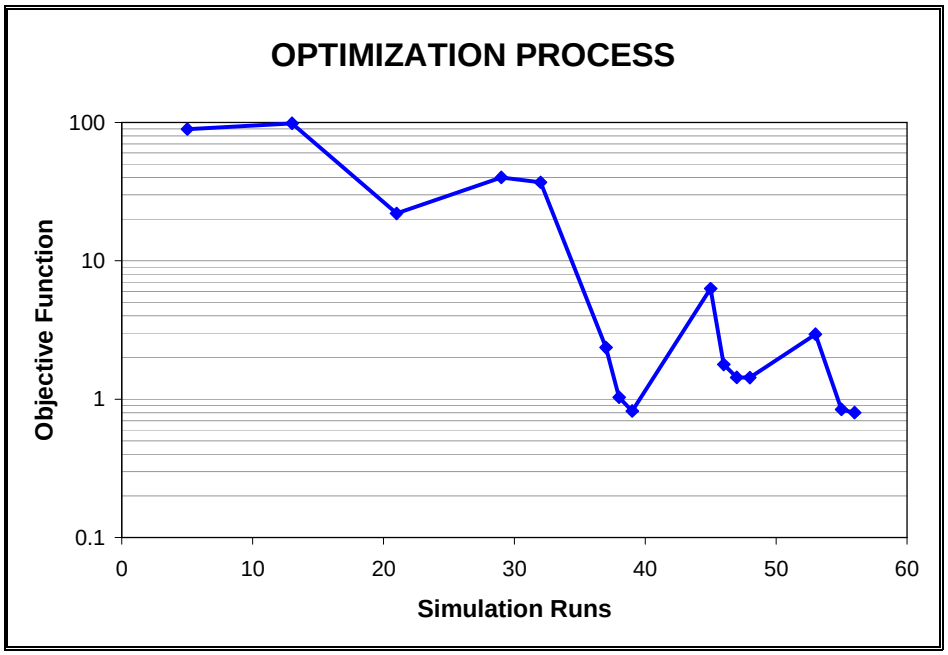


Figure 4-12. Convergence of the objective function in the validation case.

5 PORE-LEVEL REPRESENTATIONS

Several properties of rocks influence the displacement of fluids through the reservoir's porous system. An accurate prediction of performance of the reservoir requires accurate depiction of flow functions and their spatial variability. In this research, it is suggested that the spatial correlation of pores and throats at the pore-level influence the flow properties at the macro-scale. Therefore, perturbation of these pore-level spatial correlation characteristics perturbs all the flow functions simultaneously and consequently has a major influence in the production response of the reservoir.

Multiphase flow through porous media is controlled by multiphase flow properties such as capillary pressure and relative permeability. These multiphase flow properties, described as functions of fluid saturations, are closely related to the rock texture and particularly to the geometry and topology of the pore space.

The fundamental geometric elements that describe the pore structure include pore body, the throat size and length, and the coordination number (connectivity). In this study, characteristic distributions and spatial correlations for the geometric elements of the pore structure are obtained from literature that summarize

microtomography studies for different rock samples (Coker and Torquato, 1995; Lindquist, 1999; Venkatarangan, 2000).

5.1 PORE SPACE DESCRIPTION

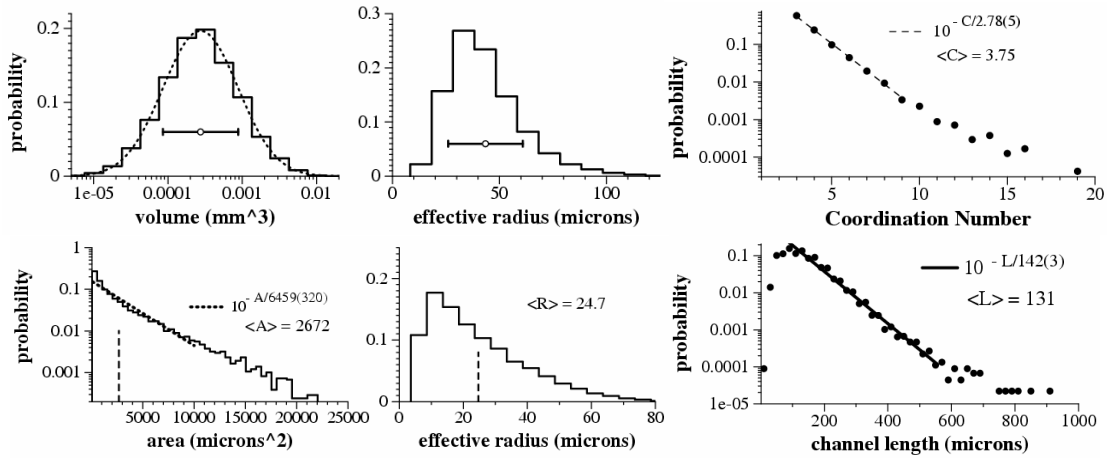
Pore-space spatial correlation has an important influence on the macroscopic flow characteristics of reservoir rocks, such as absolute permeability, residual saturation, capillary pressure and relative permeability curves. Microtomography reports include important information about the structure of the pore space, such as characteristic distributions and spatial correlation/cross-correlations for pore volumes, throat sizes, throat lengths and coordination numbers. They also describe relationships between geometric properties of the pore structure and more basic properties such as grain size distributions, sorting, porosity, cementation and compaction (Doyen, 1998; Lymberopoulous and Payatakes, 1992; Zhu et al., 1995; Lindquist and Venkatarangan, 2000; Peth et al., 2006).

Tomographic images of siliciclastic rock samples have been analyzed with geometrical interpretation algorithms to compute distributions and correlations of geometric properties in pore structures. Results from these studies indicate that distributions of pore volumes can be described as log-normal. The average and standard deviation of the pore volume log-normal distribution is correlated to the

mean and sorting characteristic of the grain size distribution. The remaining properties, including coordination number, throat length and throat area, exhibit distributions that can be well characterized by an exponential model of the form $P(A) \propto 10^{-A/\lambda_A}$. This model is not bounded between 0 and 1, and therefore requires normalization. The characteristic value of the distribution, λ_A, shows a strong correlation with porosity (see Figure 5-2). The characteristic values are directly correlated to porosity for the coordination number and throat area distributions, and inversely correlated for the throat length distribution. Consequently, the average connectivity and throat area increase with porosity, while the average throat length decreases. Furthermore, the ranges of the distribution of pore bodies, throat areas and throat lengths depend directly on the average and standard deviation of the grain size distribution.

Figure 5-1 shows some examples of measured distributions of pore volume, coordination number, throat length and throat area obtained from tomographic images of core samples from Fontainebleau (top) and Berea sandstones (bottom) (from Venkatarangan, 2000). Figure 5-2 shows the correlation between the characteristic values for distributions of geometric properties and porosity, reported in the same study. Typical characteristic values for distributions of pore structure components can be observed in these Figures.

Fountainebleau Sandstone



Berea Sandstone

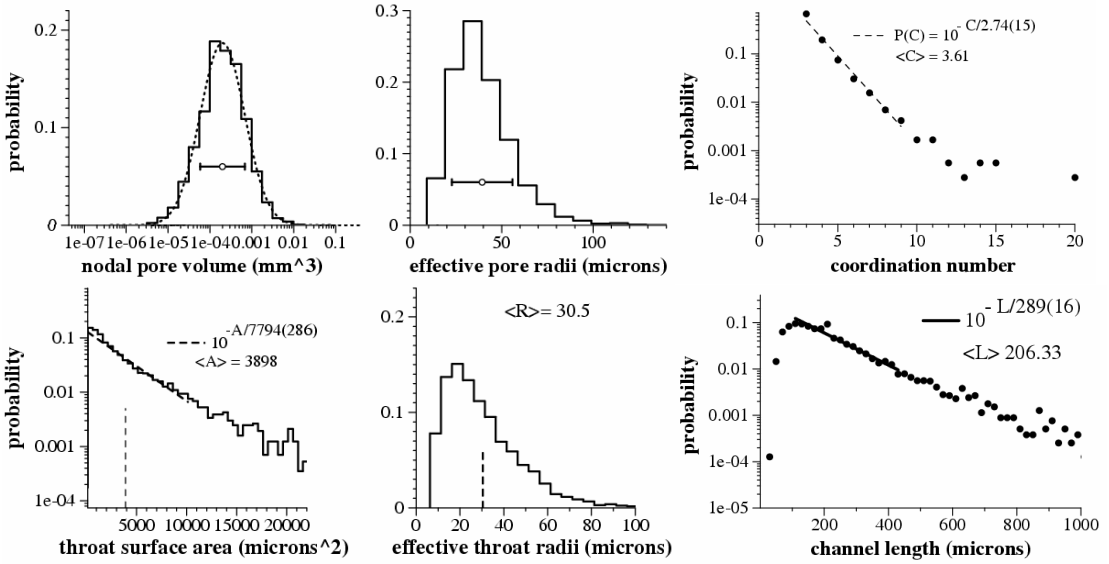


Figure 5-1. Characteristic distributions for elements of pore structure computed from tomographic images of Fountainebleau (top) and Berea (bottom) sandstone samples (reproduced from: Venkatarangan, 2000).

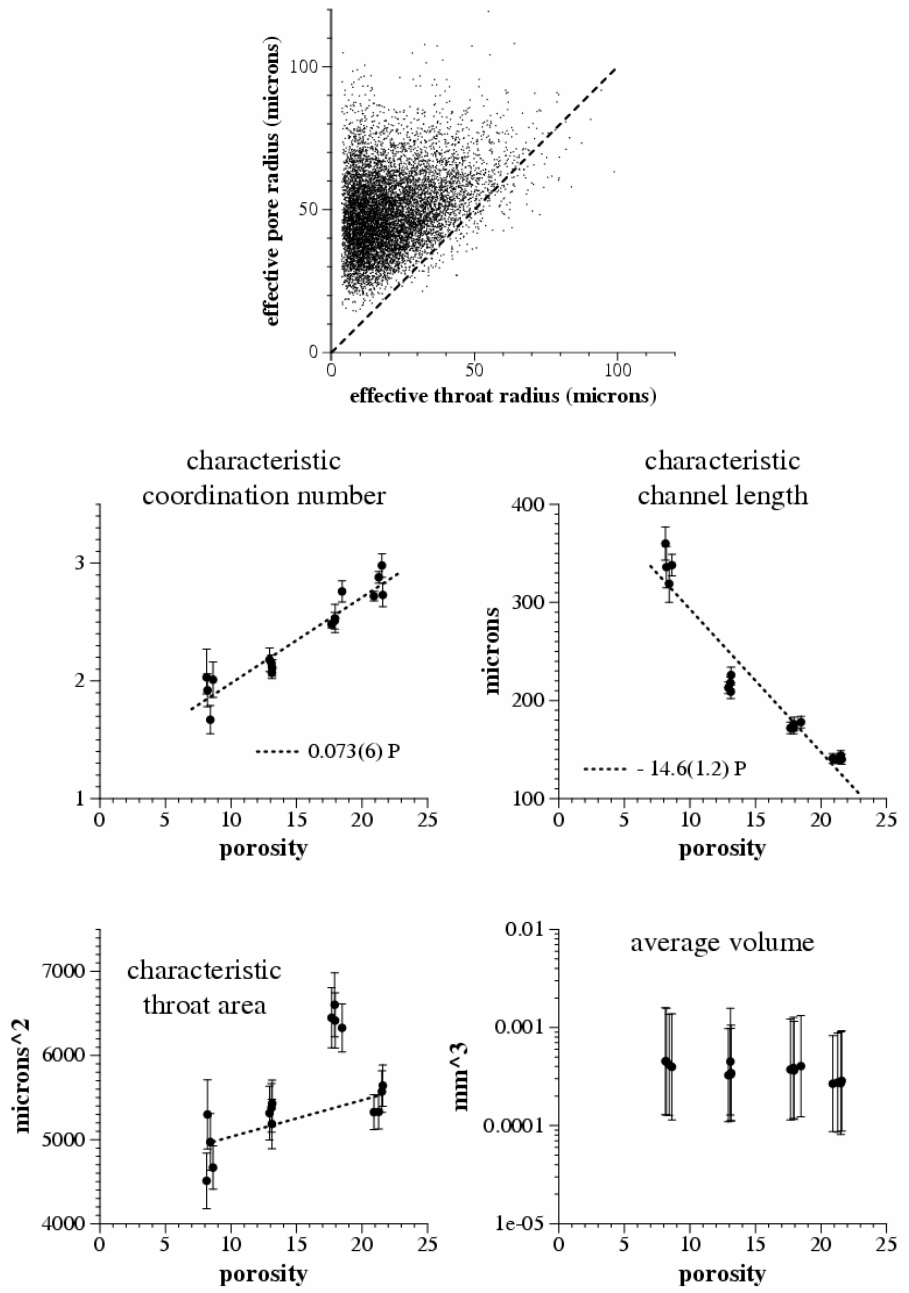


Figure 5-2. Pore-body throat-area cross-plot and; correlations between porosity and characteristic values for distributions of pore structure properties. Obtained from microtomography studies of Fountainebleau and Berea sandstone samples (reproduced from: Venkatarangan, 2000).

Summarizing, studies from microtomography images of siliciclastic rocks suggest that:

- Pore volumes generally exhibit a log normal distribution.
- The distribution of throats and connectivity is exponential.
- The magnitude of the average throat length should be comparable to the average grain size.
- The average pore body size should be close to 1/5 of the same average grain size.
- Pore body size should be directly correlated with the throat size and inversely correlated with throat length.
- The porosity should be directly correlated with the coordination number and inversely correlated with the throat length.
- The distribution of the coordination number should be exponential with a range between 2 and 10 and an average between 3 and 4.

5.2 PORE NETWORK MODEL

Percolation and pore network models have been used in the literature to model the important mechanisms involved in multiphase flow through porous media and evaluate the effect of pore-space spatial correlations on two-phase flow characteristics, during both primary drainage and imbibition.

A three-dimensional cubic lattice can be assumed to describe the pore space with pore bodies forming the nodes, inter-connected by pore throats. The size of pore bodies and throats can be modeled as probability distributions, while their allocation in the lattice can be specified considering particular models of body-body, throat-throat and body-throat spatial correlations. In this study, the spatial correlation of bodies is assumed and the spatial correlation of throats is derived by assuming a correlation between bodies and throats.

Based on microtomography studies, characteristic distributions and correlation models for geometric elements of pore structure were defined and coded into the algorithm that generates pore network models. Figure 5-3 and Figure 5-4 show some characteristic distributions and correlations for geometric elements of pore structure computed with the pore network model algorithm.

The proposed pore network model is based on a lattice where the original connectivity (number of throats connected to a single node or coordination number) was modified to obtain a variable connectivity through the lattice, generating a more realistic representation of the pore space. The maximum connectivity of the 3 dimensional cubic lattice is 18. However, most of the 18 throats for each pore body will be blocked based on the local coordination number sampled from the exponential distribution for connectivity shown in Figure 5-3. The size of the pore bodies are

sampled from characteristic distributions, shown also in Figure 5-3, based on fundamental properties of sedimentary rocks such as grain size distribution, compaction and cementation.

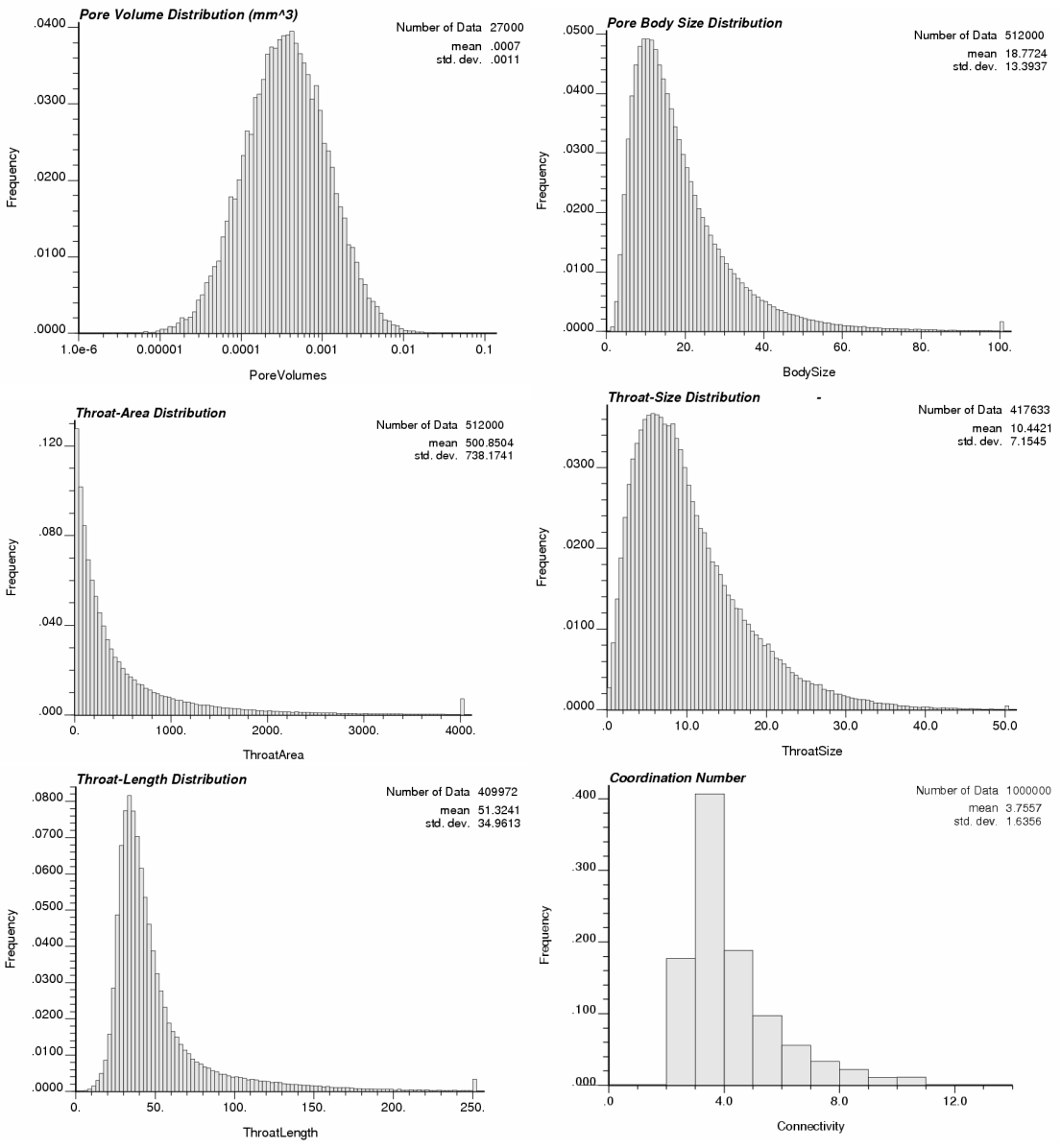


Figure 5-3. Characteristic distributions for pore body sizes, throat sizes, throat lengths and coordination number.

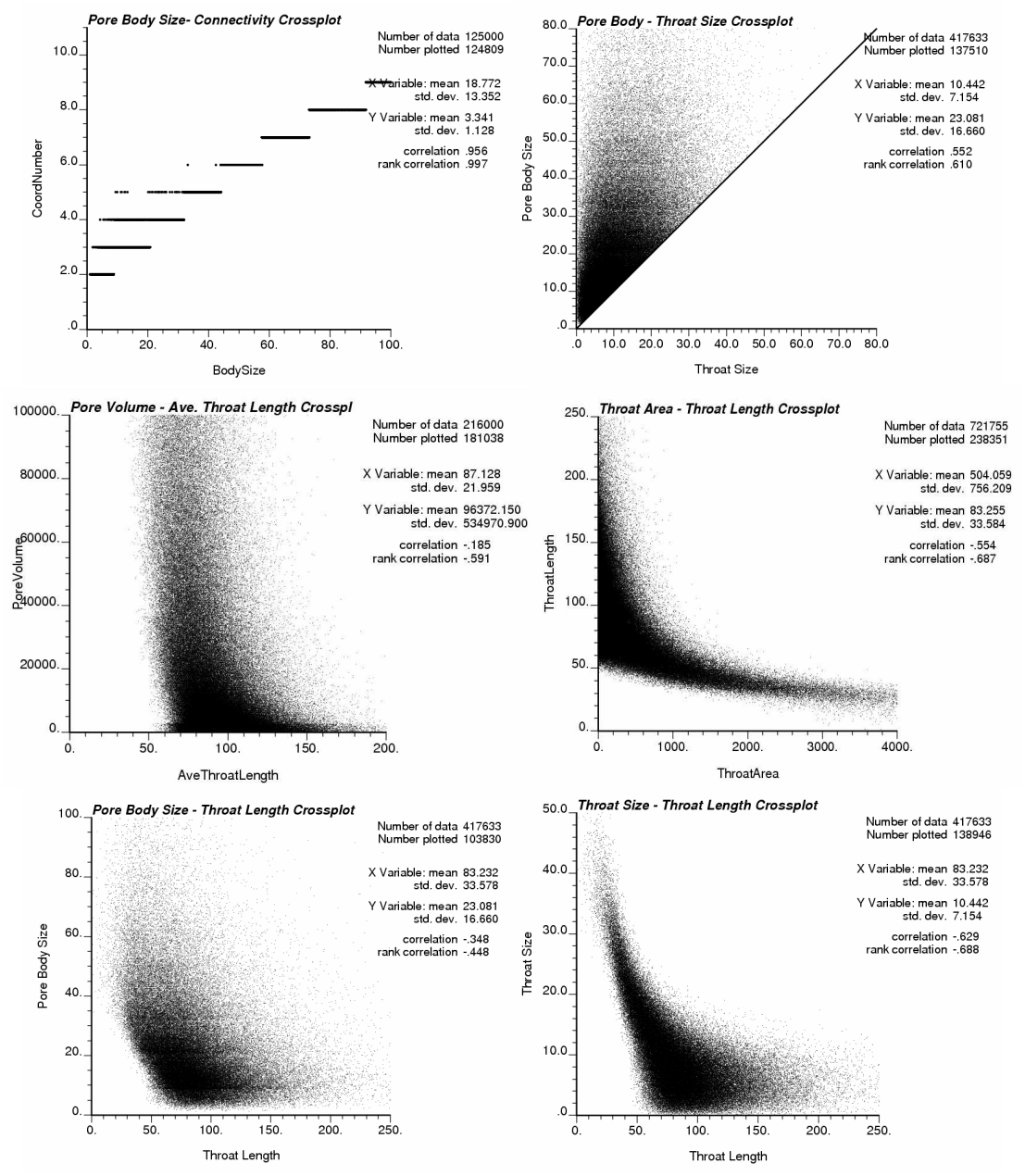


Figure 5-4. Characteristic correlations between pore body size, coordination number, throat size and throat length, based on the implemented pore network model that are consistent with the observations in microtomography reports (Venkatarangan, 2000).

The spatial distribution of the pore bodies is modeled using sequential Gaussian simulation to introduce spatial continuity. Similarly, other structural properties of the pore network such as the throat aperture, length, and the coordination number, are sampled from characteristic distributions that are consistent with the pore size distribution, compaction – porosity and cementation. The throat characteristics are allocated based on correlation with pore body sizes, as shown in Figure 5-4. Figure 5-5 shows 2-D slices from the spatial distributions for coordination number, pore body, average throat size and average throat length.

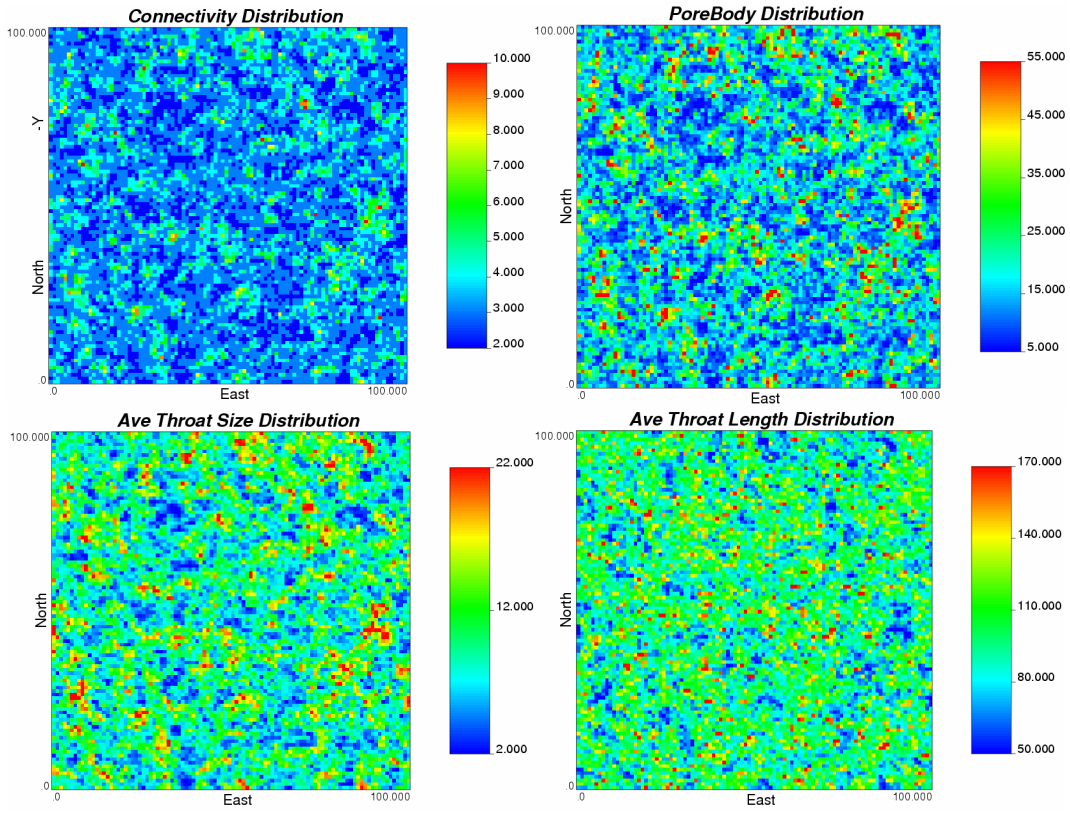


Figure 5-5. Spatial distribution of coordination number, pore body, average throat size and average throat length connected to each pore body, generated with the pore network model algorithm. Sizes are in microns. A slice through the 3-D model is shown for each attribute.

The cross-section of pore bodies and throats are irregular, bounded by nooks and crannies. Therefore, the sizes drawn from the characteristic distributions correspond to the largest inscribed circles (for throats) or spheres (for pore bodies) encompassing the irregular pore elements. A “roughness volume fraction” of 0.2 is introduced to consider the fraction of the total pore volume included in these nooks and crannies. The wetting phase is present in the roughness volume fraction even when the pore body or throat is invaded by the non-wetting phase. The fraction of the roughness pore space occupied by the wetting phase is a function of the capillary pressure and the interfacial tension between the wetting and non-wetting phases.

In the proposed approach, pore network models are calibrated by modifying basic rock properties such as average and standard deviation of grain sizes, porosity and cementation. These properties form the basis to compute the probability distributions, spatial correlations and cross-correlations for the pore-level elements (pore bodies, throats and connectivity). In turn, the resulting pore networks are used to compute the basic flow functions that will be used, after proper upscaling, in the history matching process.

5.3 PORE NETWORK SIMULATOR

Pore-level network simulation is used to evaluate the impact of pore-space structure and spatial correlations on macroscopic multiphase flow properties. Capillary, viscous or mixed controlled displacement mechanisms can be modeled in pore network models. Viscous and mixed forces become important close to the wells where fluid velocity is higher, while capillary forces control the fluid displacement in regions away from the wells. Our approach assumes capillary control and no-gravity effects for both two-phase displacement mechanisms, primary drainage and imbibition, at pore level.

Capillary control or capillary-dominated assumption is equivalent to a quasi-static displacement where the migration of fluid within the pore-space is independent of viscous forces. The amount of capillary-dominated flow is measured by the viscous-capillary number N_{vc}, a dimensionless parameter defined as the ratio of viscous-to-capillary forces:

$$N_{vc} = \frac{\mu u}{\sigma} \tag{5.1}$$

where μ is the fluid viscosity, u is the interstitial velocity, and σ is the interfacial tension. Based on literature that validates pore network simulation results

against experimental data, we can assume that the pore-network simulation results are valid for reservoirs with low viscous-capillary numbers ($N_{vc} < 10^{-5}$), where capillary forces are dominant.

Under the capillary control assumption, the resulting multiphase functions are independent of the pressure gradient. Different mechanisms can be modeled in the pore network simulation depending on the type of displacement (drainage or imbibition). These mechanisms define the invasion scheme for each displacement and are designed to mimic the real behavior of multiphase flow on porous medium. The results of this simulation are the capillary pressure and relative permeability curves. The invasion scheme is based on the work from Mani and Mohanty (1999). The pore network simulator was developed by coupling an invasion algorithm with the pore network model, to estimate capillary-controlled two-phase relative permeability and capillary pressure curves in strongly water-wet systems. These results are considered to be valid at very low capillary numbers. Primary drainage and imbibition displacements were modeled in the invasion algorithm, considering different pore level mechanisms. The pore network algorithm and an example of an imput file are presented in Appendices A and B, respectively.

5.3.1 PRIMARY DRAINAGE

During primary drainage, a non-wetting fluid (oil) invades a medium saturated with wetting fluid (water). In this displacement, the capillary pressure between invading non-wetting fluid and the resident wetting fluid is gradually increased. The invasion process is controlled by the throat sizes. However, both the throats and pore bodies contribute to the fluid saturation at any capillary pressure.

A throat containing the wetting fluid can be invaded only if it is accessible to the non-wetting fluid and its size (radius) is higher than the critical value required to balance the current capillary pressure, considering the curvature of the interface (which depends on the radius of the throat) and the given interfacial tension. Thus, as the capillary pressure is gradually increased, pore throats of smaller size are invaded.

Once the pore throat is invaded, the adjacent pore body is also invaded considering that any pore body is larger than the throats connected to it. All the throats and pore bodies that can be invaded at a particular value of capillary pressure are filled with the non-wetting phase until further invasion is not possible. Therefore, this process is controlled by the size of the largest throat accessible to the non-wetting phase.

The invasion algorithm assumes that the wetting phase forms a continuous layer along the walls of the pore space and can exit the medium at sufficiently high capillary pressures. Consequently, there is no trapping of the wetting phase at high capillary pressure values. This is consistent with experimental observations of strongly water-wet media (Dullien, 1992).

Primary drainage can be stopped considering different criteria, for instance, when a particular value of water saturation (connate saturation) is reach, when the capillary pressure arrives to a maximum value, or when a particular fraction of the pore bodies are filled with the non-wetting phase (oil).

5.3.2 IMBIBITION

During the imbibition, the final capillary pressure value obtained at the end of the primary drainage is gradually decreased to allow the displacement of the non-wetting fluid (oil) by the wetting fluid (water). During imbibition, the wetting phase preferentially imbibes into smaller pore bodies and throats.

Experiments show that two different mechanisms dominate during imbibition: i) piston-like displacement and ii) snap-off. In piston-like displacement, a pore body can be invaded by the wetting phase at a particular capillary pressure value, only if its

apparent size is lower than the critical value required to balance the current capillary pressure. An apparent pore body size is introduced based on experimental observations that show that the probability of invasion of a pore body by the wetting phase decreases with the number of connected throats filled with non-wetting phase. The apparent size of a pore body is a function of the radius and the number of throats containing the non-wetting fluid. Once a pore body is invaded, all the connected throats containing the non-wetting phase are invaded.

In the snap-off mechanism, the size of a throat containing non-wetting fluid is sufficiently small to suck in the wetting fluid from adjacent throats, bypassing the non-wetting phase residing within the pore body.

Both pore level mechanisms can take place only if the pore body containing the non-wetting fluid is part of a connected path to the outlet boundary of the pore network. These mechanisms lead to the generation of blobs or disconnected volumes of the non-wetting phase. Consequently, as the imbibition process proceeds, a point is reached where further reductions in the capillary pressure value do not affect the fluid saturation. At that point (end of imbibition) the saturation of the non-wetting phase is the residual saturation.

5.3.3 CAPILLARY PRESSURE CURVES

The invasion algorithm considers capillary controlled displacements. During primary drainage, the gradual increment in the capillary pressure causes the progressive invasion of the non-wetting fluid in the pore network model, initially saturated with the wetting phase. The invasion at a particular capillary pressure value proceeds until the largest throat accessible to the non-wetting fluid is small enough to balance the capillary forces and stop the invasion. At every capillary pressure value the saturation of the non-wetting phase is calculated considering the volume of all the invaded throats and pore bodies. Combining the results for all the capillary pressure values, the relationship between the capillary pressure and the saturation state (or primary drainage capillary pressure curve), is obtained.

Similarly, during imbibition, the gradual reduction of capillary pressure causes the invasion of the wetting fluid back in to the pore network model while the non-wetting fluid recedes. The saturation state at the end of the primary drainage is the initial state for the imbibition. For a particular value of capillary pressure, the invasion of the wetting phase proceeds until either the smallest pore body accessible to the wetting phase is large enough to balance the capillary forces or the smallest adjacent throat is large enough to avoid a by-pass of the wetting fluid or snap off. In both cases, a connected path of the non-wetting fluid to the outlet side of the model is

required for the invasion to take place. Once more, the compiled results for all the capillary pressure values define the relationship between the capillary pressure and the saturation state during the imbibition. (or imbibition capillary pressure curve)

5.3.4 RELATIVE PERMEABILITY CURVES.

The phase distributions in the pore network model at each capillary pressure are equivalent to those from a steady-state two-phase flow experiment. For each phase, continuous path from the inlet to the outlet boundaries of the pore network model, are identified. A small pressure drop is imposed in the flow direction for the pore bodies and throats that form the continuous path of each phase.

The flow through individual throats is modeled with the Hagen-Poiseuille equation for steady-state flow in tubes, considering the viscosity of the phase and the dimensions of the throat. The algorithm assumes that after the throat invasion, the flow within each phase quickly reaches steady-state and can be modeled with Hagen-Poiseuille equation. The magnitude of the pressure drop across the pore bodies is much smaller than that across pore throats, and consequently can be ignored in the calculations. The nodal pressures are solved from a system of equations considering mass balance across the pore bodies.

$$q_i = \frac{\pi \, \Delta P_i \, r_t^4}{8 \, \mu_i \, l}; \quad i = o, w \tag{5.2}$$

The exact Hagen-Poiseuille equation is applied only when the throat and the adjacent pore bodies are occupied by a single phase. This equation is modified under two conditions. The first condition applies for both phases when either one or both of the adjacent pore bodies contain a different phase from that residing in the throat. The second condition applies only for the wetting phase when most of the throat is occupied by the non-wetting phase and the wetting phase is present only in the surface roughness volume. The throat size in the calculation of the throat conductivity is reduced by a factor of 1.5 and 10 for the two conditions, respectively, assuming that is equivalent to the transport of fluids in smaller tubes (Mani and Mohanty, 1999).

The imposed pressure drop and the calculated inlet flow rate of each phase are combined with Darcy's law to compute the phase relative permeability corresponding to each capillary pressure/saturation state. The absolute permeability is calculated considering a pore network model saturated with a single phase. Figure 5-6 shows results of the invasion algorithm for a particular saturation state.

Once the multiphase flow macroscopic properties are determined by the pore network simulation, they are used after upscaling as input functions for the flow simulation. That way, the pore space spatial correlations has an impact in the flow simulation, by altering the multiphase flow characteristics. The spatial distribution of

rock types characterized by different spatial correlations of pore space, is accomplished using the indicator simulation method.

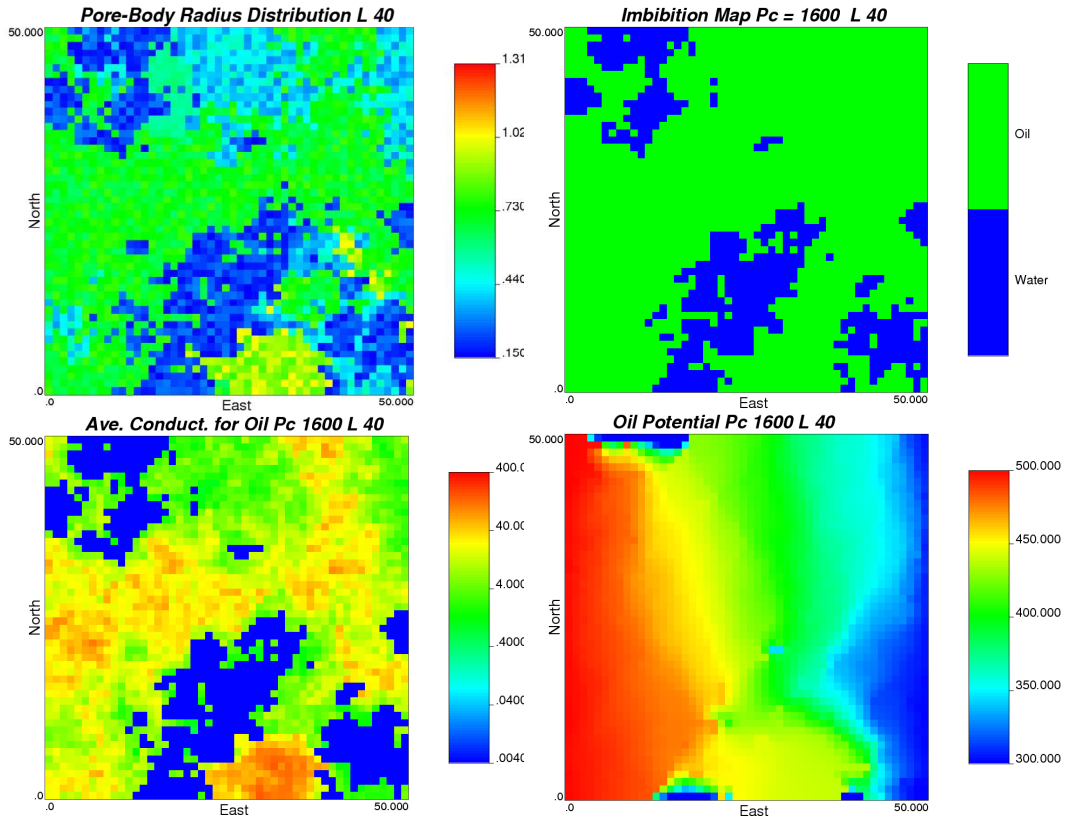


Figure 5-6. Results from the invasion algorithm including the pore-body and phase distributions in the top, and the corresponding oil conductivity and potential distributions (bottom).

By implementing a systematic method for altering the pore level spatial correlation, a new approach to fit the simulation results to the production history can be obtained. This method has the advantage of considering the relationship between

multiphase flow characteristics and spatial correlation starting from the pore level. By seamlessly transitioning from the pore scale to the grid block scale, a systematic multi-scale method for history matching can be formulated.

6 CALIBRATION OF NETWORK RESULTS

Initially, different sensitivity studies were performed with the pore network simulator to estimate the influence of different variables describing the pore structure on the resulting multiphase flow functions. Utilizing the probability distributions and correlations reported in microtomography studies of rock samples, a representative pore network model can be built based on few basic parameters. Typical parameters obtained from analysis of core samples include the average grain size and sorting, porosity and/or compaction and the cementation level. The model size is determined by the number of pore bodies. However, the actual length of the model also depends on properties such as average pore body size and throat length. A model with a particular number of pore bodies can expand or contract to reproduce the porosity and the correlation model for the spatial distribution of body size and throat length.

Based on the sequential simulation approach, multiple equiprobable realizations for the distribution of pore bodies can be generated. Depending on the size of the model compared to the spatial correlation length specified, spurious fluctuations in the network characteristics may be observed due to non-ergodicity. To capture only the effect of spatial correlation on the network characteristics, the minimum model size required to obtain an acceptable reproducibility of the resulting

multiphase flow functions was ascertained. Models with a number of pore bodies ranging from 8000 to 1’000000 were evaluated.

Other sensitivity studies evaluate the variations of static properties and flow functions with elements of pore structure and rock texture. Pore structure properties such as average pore body size, average throat size and length, connectivity; and multiphase flow properties such as absolute permeability, relative permeability end points, oil residual saturation, and capillary pressure curves were evaluated in these studies. Additional studies to evaluate the influence of layered systems and rocks with multiple sediment sources on multiphase flow functions were also performed. Finally, studies for scaling multiphase flow functions by coupling pore networks and flow simulation results were pursued.

6.1 EFFECT OF PORE NETWORK MODEL SIZE

Since a stochastic approach was undertaken to represent the pore structure, multiple realizations of the pore network were generated. It was observed that the calculated multiphase flow functions over multiple realizations of the network merged as the model size increased. Multiphase flow functions are case or realization dependent for small pore network models; consequently, a minimum model size is required to obtain an acceptable reproducibility in the results such that the resultant

functions only represent the influence of pore level spatial correlations and not the effect of non-ergodicity (Figure 6-1). This minimum model size depends on the spatial correlation of the pore body sizes (normally between 2 and 10 pore bodies) and the grain size sorting. Rocks with poorly sorted grain size distribution require bigger models to reflect reproducible results. For the study cases, the minimum model size was between 100000 and 200000 pore bodies. However, the minimum model size also depends on the specified spatial correlation and can not be generalized.

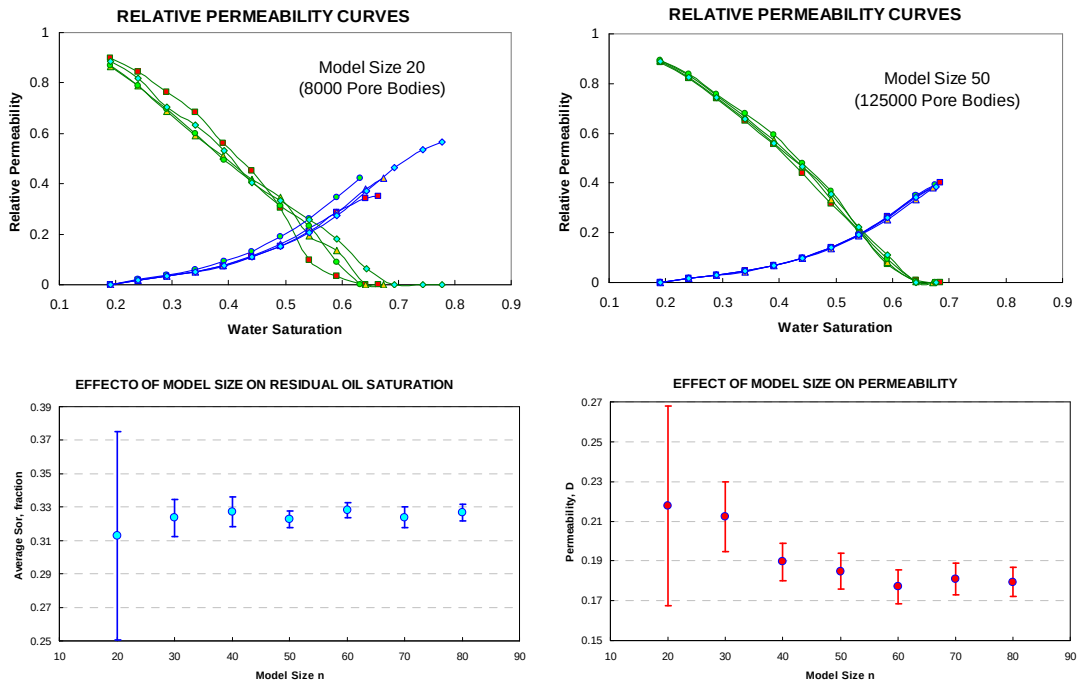


Figure 6-1. The influence of pore network model size on the computed pore network properties and multiphase flow functions. These results are based on 7 realizations of the network for each model size.

6.2 EFFECT OF SPATIAL CORRELATION

The impact of pore level spatial correlations on the macroscopic petrophysical properties and multiphase flow functions is evaluated next. Pore network models with the same grain size distribution and porosity, but different spatial correlation of pore body size were evaluated with the invasion algorithm. Other elements of the pore network model, such as coordination number and throat size and length, are directly or inversely correlate to the pore body size. Consequently the allocation and spatial correlation of other pore network elements is directly affected by spatial correlation of the pore body size. Figure 6-2 shows the spatial distribution of pore body sizes and the single phase pressure distribution for two models with identical grain size distribution and porosity but different pore body spatial correlation. The single phase pressure distribution is estimated with the invasion algorithm to determine the total flux and the absolute permeability of the porous medium. It can be noticed that the pressure solution for short-correlation pore network models exhibit a more gradual transition between the boundary values of the model.

The effect of pore model size on the calculated static and multiphase flow properties increases with spatial correlation. As the spatial correlation increases the results become more realization dependent and bigger pore network models are required to ensure reliability and reproducibility in the results. To reduce the impact

of the model size on the results of the invasion algorithm, large pore network models (512000 pore bodies) were used in this study.

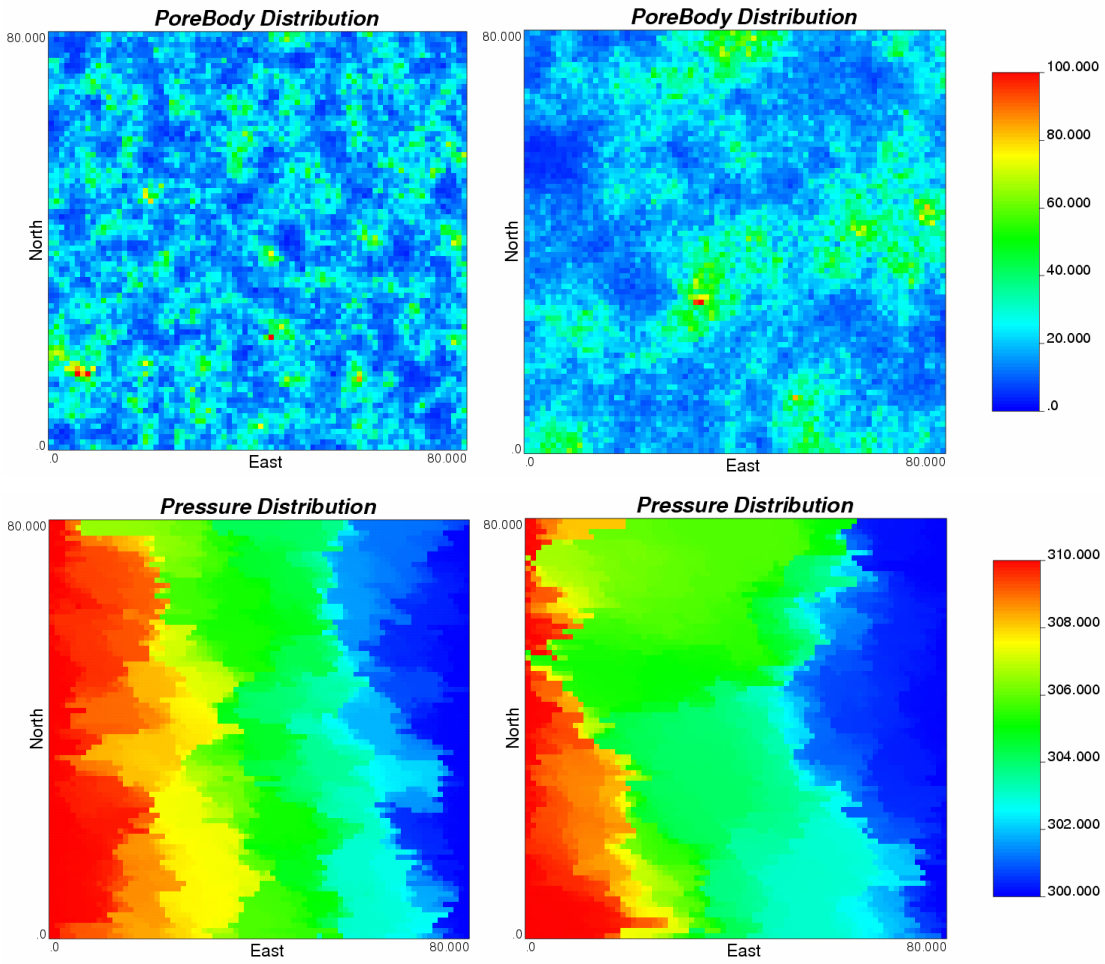


Figure 6-2. Spatial distribution of pore bodies (top) and single phase pressure solution (bottom) for 3D pore network models with short (left) and long (right) spatial correlation (4 and 16 pore-body correlation lengths). Pore bodies are in microns and pressure in psi.

The impact of the spatial correlation on the absolute permeability and multiphase flow properties such as residual saturations and relative permeability end points, were analyzed. Results from the study confirmed that the absolute permeability is directly correlated to the spatial correlation (as expected) (see top left plot in Figure 6-3). These results are based on single realizations of the network models, whose size is big enough to not be unduly affected by ergodic fluctuations. An increment in the spatial correlation of the pore bodies, along with the associated changes in connectivity and throat properties, favor the presence of high conductivity paths through the pore network model, increasing the absolute permeability (see Figure 6-2). Even though the probability of low conductivity paths also increase, the absolute permeability is highly influenced by the best conductivity paths.

In water wet porous media, residual oil is normally trapped in large pore bodies and throats under capillary controlled displacements. This is not necessarily true when viscous forces are taken into consideration. Results of the study show that the oil residual saturation remains relatively constant (with an initial gentle raise) with increasing spatial correlation of the pore bodies (see top right plot in Figure 6-3). However, the distribution of the residual oil in the model changes dramatically with the spatial correlation. The size of the trapped oil blobs representing the residual oil increases with the spatial correlation, at the same time as the number of blobs decreases (see Figure 6-4). Changes in the size and the number of oil blobs balance each other to maintain the oil residual saturation relatively constant. However the

small rise in the oil residual saturation with low values of spatial correlation suggests that the increment in size of the oil blobs overcomes their reduction in number at low values of pore body spatial correlation.

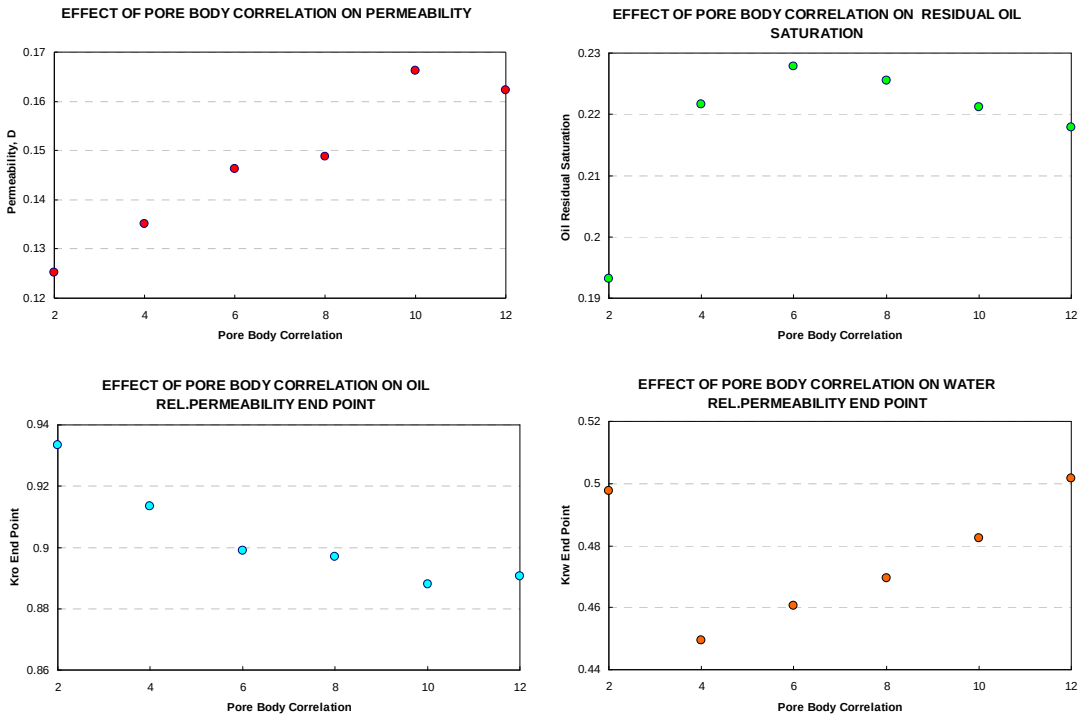


Figure 6-3. Results of the sensitivity study for the influence of spatial correlation in the computed pore network properties and multiphase flow functions.

The behavior of the water relative permeability end point with the spatial correlation is consistent with the oil residual saturation (see bottom right plot on Figure 6-3). In capillary control led displacement through water wet rock, connate water occupies small pore bodies and throats. The distribution of connate water in the pore network model at the end of primary drainage exhibits the same behavior of the

residual oil with the spatial correlation at the end of imbibition. The connate water accumulates in fewer and bigger volumes in models with higher spatial correlation (see Figure 6-4). The oil relative permeability end point gently decreases with spatial correlation (see bottom left plot on Figure 6-3), however, the oil effective permeability still increases considering the more evident increment of absolute permeability with the spatial correlation.

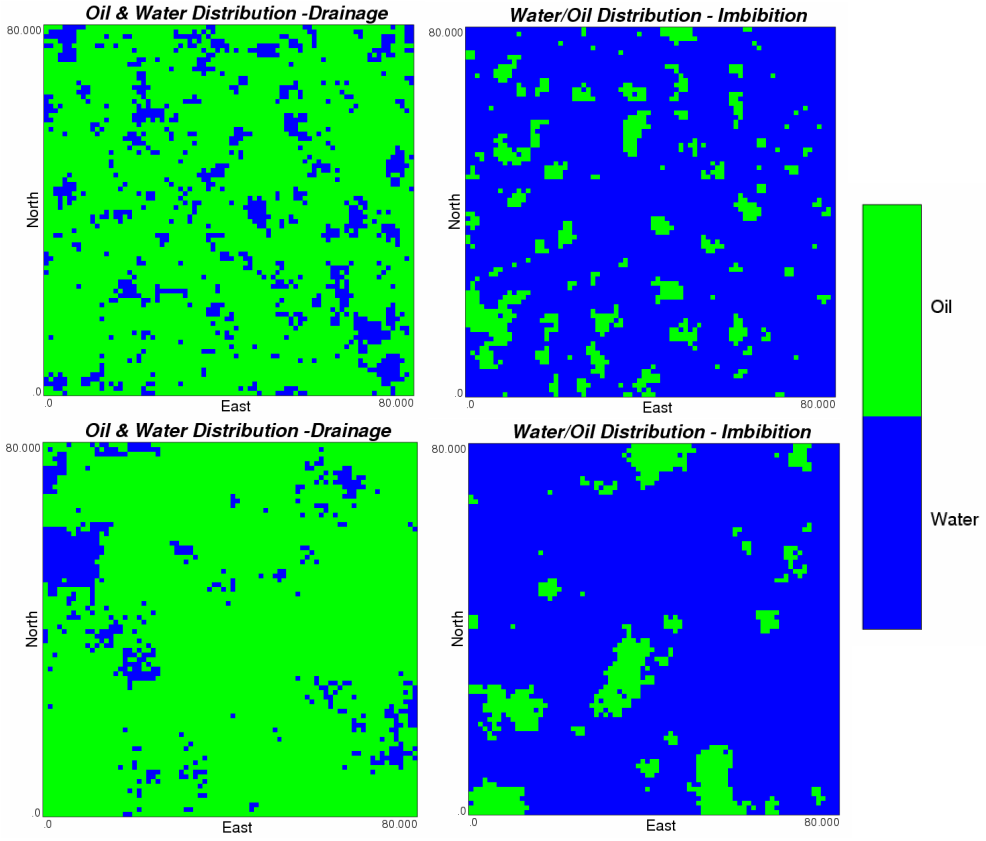


Figure 6-4. The distribution of phases at the end of primary drainage (left) and imbibition (right) for 3D pore network models with short (top) and long (bottom) spatial correlation.

Considering that the value of connate water saturation is similar for all models, oil relative permeability end point values seem to be primarily affected by the size of the connate water blobs. The oil relative permeability end point gradually drops as spatial correlation and the size of the corresponding connate water blobs increase. On the other side, additional factors affect the relationship between water relative permeability end point and spatial correlation. The water relative permeability end point is affected, not only by the size and number of residual oil blobs, but also by changes in residual oil saturation and dominant displacement mechanism during imbibition, i.e. piston like displacement or snap off. The initial drop in the water relative permeability end point can be intuitively explained by the corresponding jump in residual oil saturation. However, residual oil saturation remains relatively constant at higher spatial correlation values and consequently, other factors come into play in determining water relative permeability end points.

Two possible explanations for the gradual increase in water relative permeability end point at higher values of spatial correlation could be, i) the effect of a decreasing number of residual oil blobs overcomes the effect of an increase in their size in determining water relative permeability end point value, and/or ii) the dominance of snap off displacement mechanism during imbibition is notably reduced with an increasing spatial correlation, thus reducing the number small oil blobs trapped in single pore bodies and throats, improving the global conductivity of the water phase. Discarding ergodicity, fluctuations in the estimated water relative

permeability end point are caused by changes in the relative influence of the multiple factors presented above. Relative permeability end points of oil phase compared to - water phase exhibit more gradual changes due to the reduced number of factors affecting its estimation.

6.3 POROSITY EFFECT

In the porosity sensitivity study, multiphase flow functions and static properties calculated from multiple pore network models with the same grain size distribution and sorting, but different porosity were evaluated. Some important observations include an exponential relationship between porosity and permeability; and an increase in residual oil saturation (accompanied by a decreasing oil relative permeability end point) with porosity (See Figure 6-5). The results can be explained by the direct relation between porosity, connectivity (coordination number) and throat size; and the inverse relationship between porosity and throat length. Particularly important is the direct relationship of porosity and connectivity, since higher connectivity increases the conductivity (permeability) of the pore network but at the same time results in alternative fluid flow paths that in turn leads to an increase in the number of bypassed oil blobs, resulting in higher residual oil saturation.

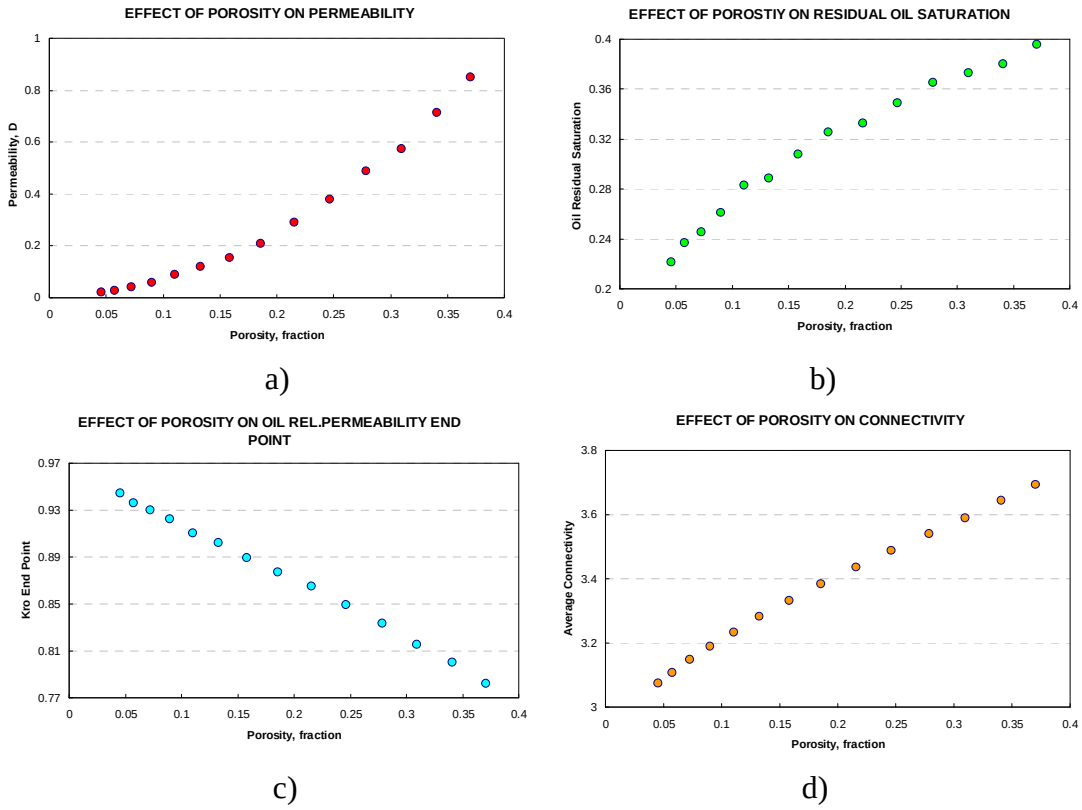


Figure 6-5. Results of the sensitivity study for the influence of porosity on the computed pore network properties and multiphase flow functions; a) Influence of porosity on absolute permeability; b) Influence of porosity on residual oil saturation; c) Variations in end-point relative permeability to oil with porosity, and; d) Effect of porosity on network connectivity.

6.4 EFFECT OF GRAIN SIZE SORTING

In the sorting sensitivity study, multiphase flow functions corresponding to pore network models with the same average grain size and porosity, but different sorting or grain size variance are evaluated. The most significant influence of

increasing grain size variance is the reduction in permeability, increase in oil residual saturation and decrease in water and oil relative permeability end points (particularly for water). These results can be observed in Figure 6-6.

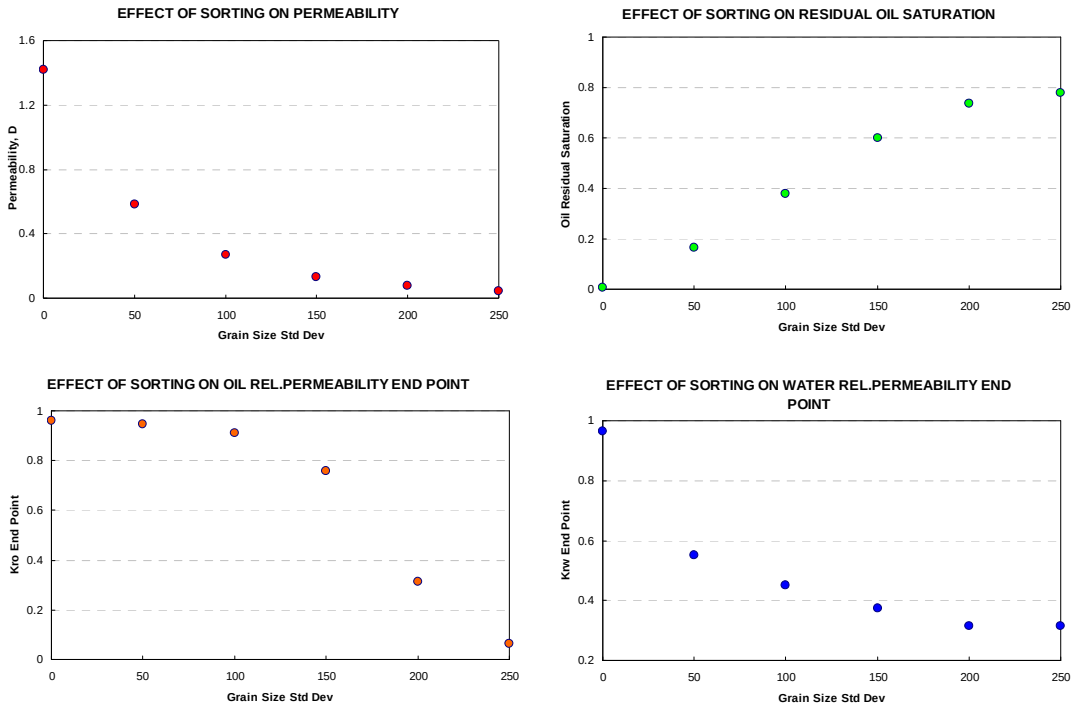


Figure 6-6. Results of the sensitivity study for the influence of grain size variance on the computed pore network properties and multiphase flow functions.

These results can be explained by the direct relationship between the variance of the grain size distribution and the connectivity of the network, as well as the variance of the pore body and throat size distributions. Even though the average connectivity remains constant, by increasing the variance of the connectivity distribution through the network, the global conductivity (permeability) is notably

reduced because of the increased proportion of poorly connected pore bodies. In addition, the number of smaller pore bodies and throat sizes also increase. Also, the increased variance of the throat size and pore body distributions favor the trapping of oil blobs since water tends to travel faster through small throats, and oil blobs tend to be bypassed and trapped in bigger pore bodies and throats (under the assumption of capillary controlled displacement).

6.5 EFFECT OF LAYERS

In reality, the reservoir rock is almost never made up of a single pure rock type (with a unique pore network). The generation of more complex models considering composite systems such as layered models and other depositional environments with multiple sediment sources is discussed in this section. The grain size distribution in these complex models is characterized by bi-modal or multi-modal distributions. Up to five different sediment sources can be considered in the work done in this dissertation. Additionally, layered systems can be oriented parallel or perpendicular to the flow. In the following sensitivity study, static properties and multiphase flow functions for a two-layer pore network model with different ratios of layer thickness are evaluated. The basic parameters that characterize each of the layers are presented in the following table.

Parameter	Layer 1	Layer 2
Average Grain Size (micro-m)	35	100
Grain size Stand. Dev. (micro-m)	60	50
Porosity (fraction)	0.12	0.18

Table 6-1. Summary of properties for two layers in a composite, two-layer network.

Different observations about the relationship between the pore structure and the static and multiphase flow properties at a continuum scale can be made from these layered models. There are two particular cases where these layered models can be useful: to study rocks with mesoscopic heterogeneity such as ripples and small-scale cross lamination and to evaluate the effect of the mixture of lithologies on the petrophysical properties and multiphase flow.

6.5.1 HORIZONTAL LAYERS

In the first part of the sensitivity study a two-horizontal-layer pore network model is evaluated. In this model the lower layer (layer 2) is considered a higher quality deposit with better transmissibility and porosity (see Figure 6-7). The thickness ratio, defined as the thickness of the layer 2 over the total thickness of the model, is varied from 0 to 1. For a thickness ratio of 0, the layer 1 (upper) occupies the entire model. Conversely, a pore network model with a thickness ratio of 1 considers only the properties of the layer 2. Figure 6-7 shows the distribution of pore

bodies, single phase pressure solution, and the phase distribution at the end of the primary drainage and imbibition displacements, for a two-horizontal-layer pore network model with thickness ratio of 0.5.

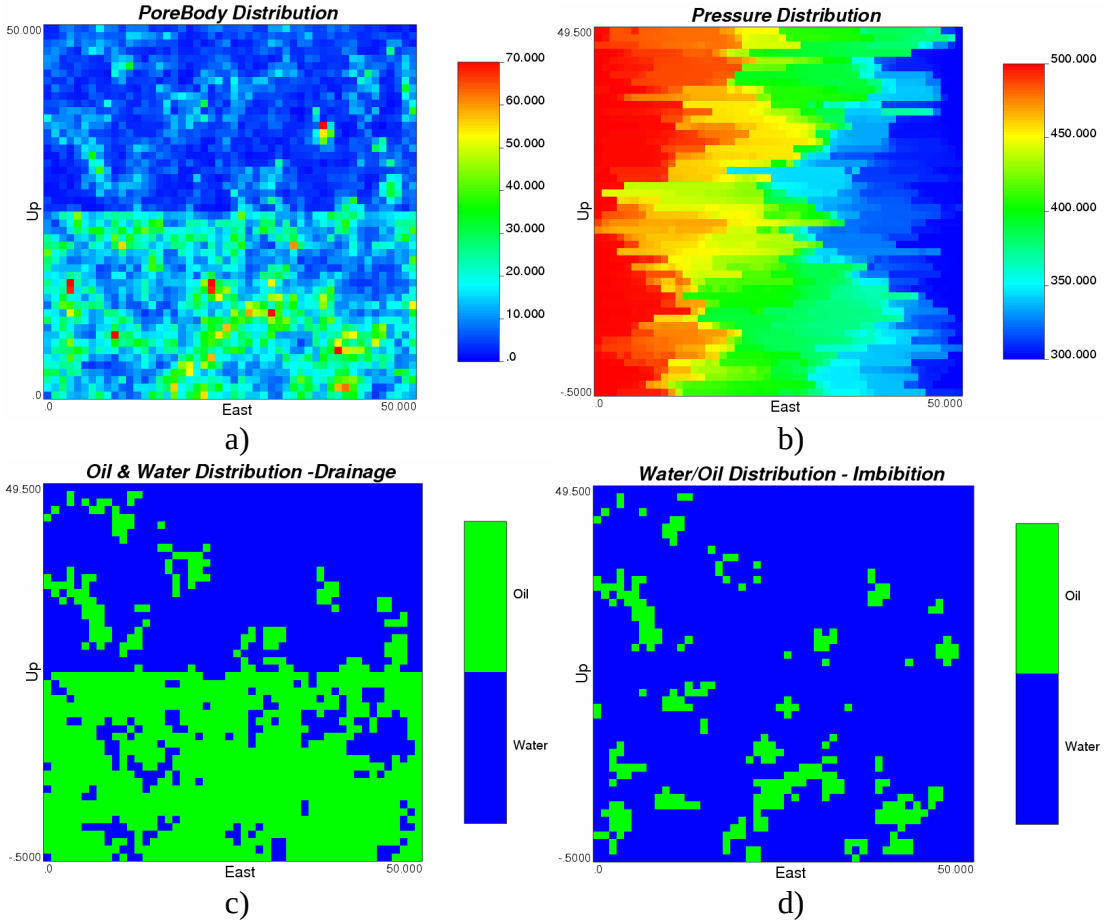


Figure 6-7. a) Pore body distribution for a model with two layers of contrasting porosity and connectivity; b) Calculated single phase pressures for the two-horizontal-layer pore network model; c) Distribution of phases at the end of primary drainage in the two-layered model; d) The distribution of phases at the end of the imbibition process in the two layered model.

For capillary controlled displacement, the distribution of connate water is associated with small pore bodies and throats. Consequently, the connate water at the end of the primary drainage is located mostly in the low quality rock (upper) which has smaller pore bodies and throats. On the other side, the residual oil at the end of a capillary controlled imbibition is mainly in the bigger pores and throats. Even though the high quality lower layer has bigger pores and throats, there is a significant fraction of residual oil trapped in the low quality layer due to the presence of isolated oil blobs without continuous oil-paths to reach the outlet boundary. The isolated oil blobs in the low quality layer are a consequence of the high connate water saturation that persists in that layer at the end of the primary drainage. Figure 6-7 shows the distribution of connate water at the end of primary drainage (bottom left) and the distribution of residual oil at the end of imbibition (bottom right).

The static and multiphase flow properties of the pore network model at different values of thickness ratio fall between the properties of the individual layers, as expected (see Figure 6-8). However, while the permeability values exhibit a smooth transition between the permeabilities of the two layers, the multiphase flow properties, including the oil residual saturation and the end point of the water and oil relative permeability curves, are particularly influenced by the high quality rock in the lower layer (see Figure 6-8). Even in models with relatively small thickness of high quality rock (low thickness ratios), the calculated multiphase flow properties of the total model are closer to the properties of the high quality layer alone.

Furthermore, the relationship between the static and multiphase flow properties in composite systems is non-linear and presumably can be evaluated at a pore level. Variations in the thickness ratio of the two-layered model induce changes in factors that determine the water relative permeability end point, such as residual oil saturation, characteristics of the distribution of residual oil blobs, prevalence of displacement mechanisms during imbibition (piston like displacement versus snap off), among others. Changes in the relative influence of these factors cause fluctuations in the water relative permeability end points determined with the network models (See Figure 6-8).

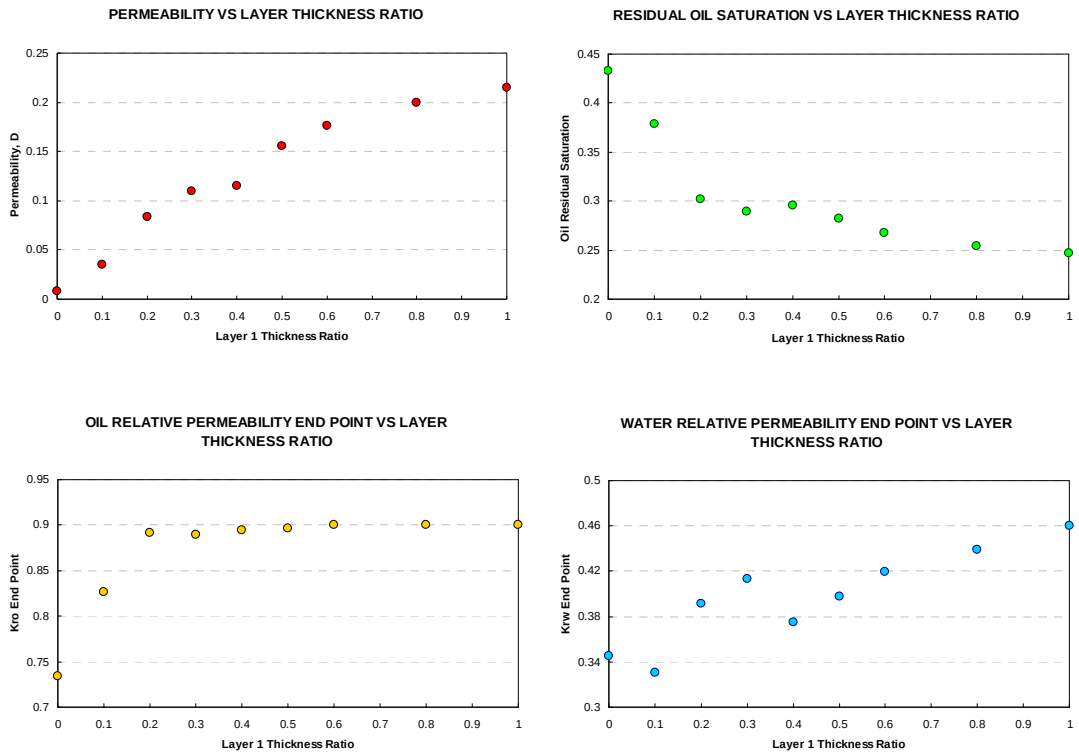


Figure 6-8. Results of the sensitivity study for the influence of the thickness ratio in the static and multiphase flow properties of a two-horizontal-layer pore network model.

6.5.2 VERTICAL LAYERS

In the second part of the sensitivity study, a two-vertical-layer pore network model is studied. In this model the higher quality deposit (layer 2) with better transmissibility and storability is located at the outlet end of the model (see Figure 6-9). The definition of the thickness ratio, the thickness of the layer 1 over the total thickness of the model, considers the vertical orientation of the layers (perpendicular to flow). For a thickness ratio of 0, the layer 2 occupies the entire model. Conversely, a pore network model with a thickness ratio of 1 considers only the properties of the layer 1. Figure 6-9 shows the distribution of pore bodies, single phase pressure solution, and the phase distribution at the end of the primary drainage and imbibition displacements, for a two-vertical-layer pore network model with thickness ratio of 0.6.

The distribution of connate water and residual oil in the two-vertical-layer pore network model show the same behavior as observed in the model with horizontal layers. Basically, distribution of the connate water is highly associated with the low quality rock while the allocation of the residual oil show a weak preference for the high quality rock. The distribution of the residual oil is significantly affected by the distribution of phases at the beginning of the imbibition. Figure 6-9 shows the

distribution of connate water at the end of primary drainage (bottom left) and the distribution of residual oil at the end of imbibition (bottom right).

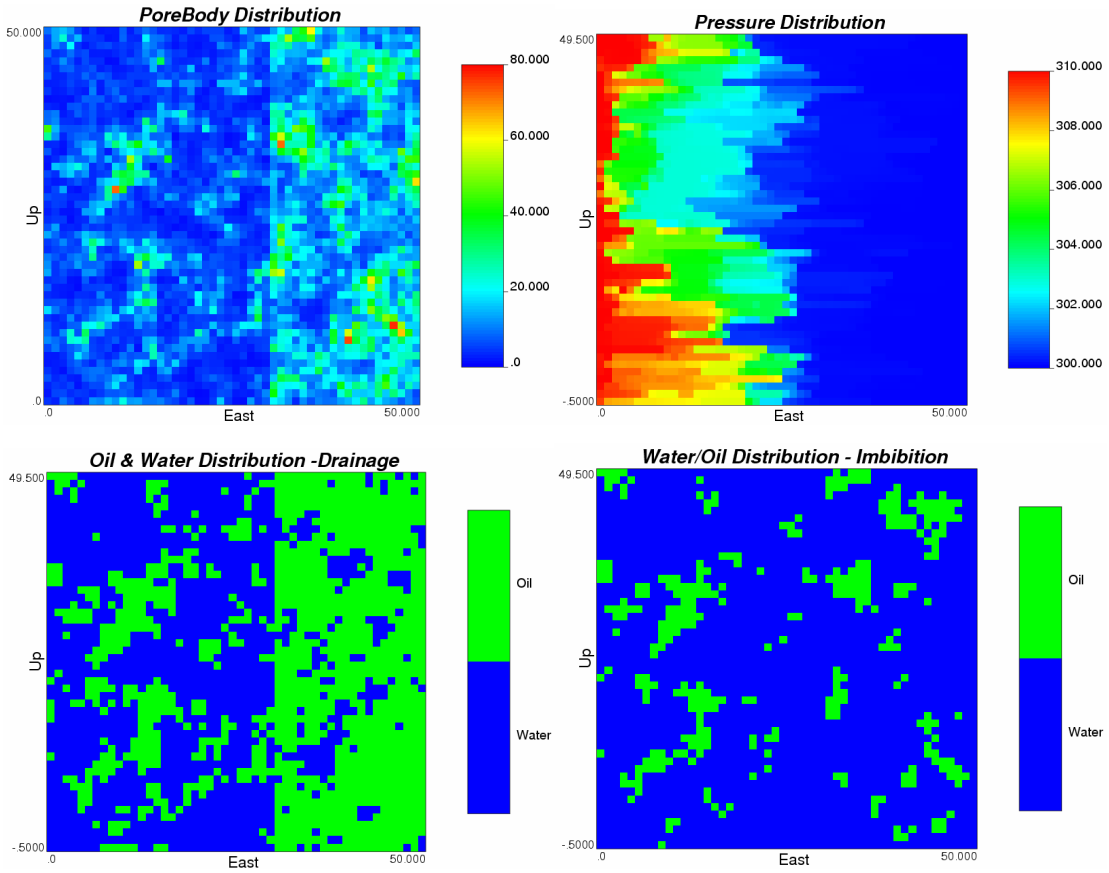


Figure 6-9. On the top: Vertical slides of the distribution pore bodies (left) and calculated single phase pressures (right) for a two-vertical-layer pore network model. On the bottom: Vertical slides of the distribution of phases at the end of the primary drainage (left) and imbibition (right) displacements for the same model. The distribution of connate water and residual oil can be identified on the bottom left and the right plots respectively.

Contrary to the case with horizontal layers, not all the static and multiphase flow properties of the pore network model at different values of thickness ratio fall between the properties of the individual layers (see Figure 6-10). Specifically, the end point relative permeability to water exhibit fluctuations that are beyond the range of variability for the pure facies. Additionally, the permeability values of the composite models don't exhibit a smooth transition between the permeabilities of the two layers, like the first case, but they proved to be highly influenced by the permeability of the low quality layer (harmonic average approx.). This is predictable considering that in this case the low quality layer occupies the entire cross section area of the flow.

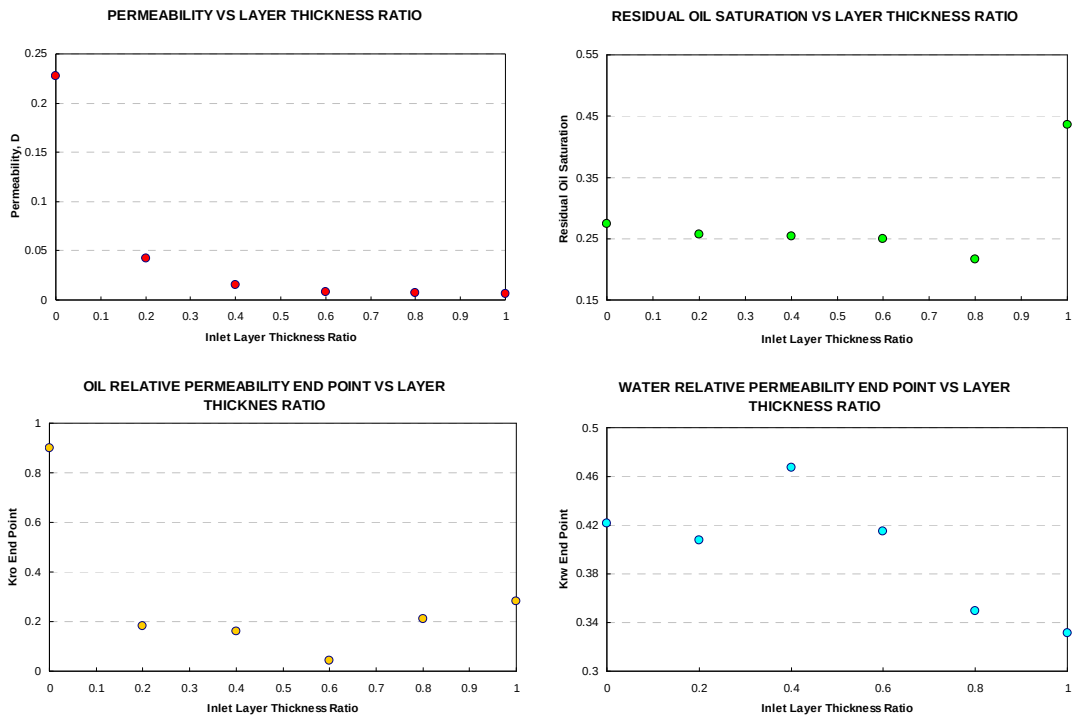


Figure 6-10. Results of the sensitivity study for the influence of the thickness ratio in the static and multiphase flow properties of a two-vertical-layer pore network model with layer oriented perpendicular to the flow.

The multiphase flow properties, including the oil residual saturation and the water and oil relative permeability end points, are particularly influenced by the low quality rock (inlet layer) (see Figure 6-10). Again, the relationship between static and multiphase flow properties in composite systems is non-linear and presumably can be evaluated at the pore level.

6.6 EFFECT OF MIXED SEDIMENT SOURCES

In the previous sensitivity study, composite pore network models based on layer systems with two different orientations (parallel and perpendicular to flow) were analyzed. In general, layered models are not representative of sedimentary rocks at the pore scale. The results nevertheless provide an indication of how the effective properties over different volume scales might get affected by transitions across different pore networks. In the next sensitivity study, assorted composite systems considering multiple sediment sources, instead of layered models, are analyzed.

An assorted, composite pore network model with two sediment sources is evaluated in this study. In this model, high quality sediments with large and well sorted grains are mixed with low quality sediments (small and poorly sorted grains) to obtain the global grain size distribution of the composite pore network model. Thus in this case we have a bimodal distribution of grain sizes originating from two different

sources. (see Figure 6-11). The basic parameters that characterize each of the sediments are presented in Table 6-2.

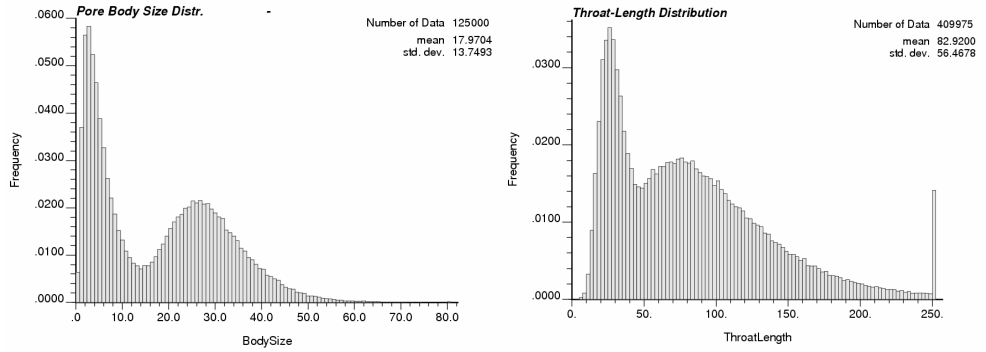


Figure 6-11. Characteristic bimodal distributions of pore body sizes and throat lengths for a pore network model composed of grain sizes corresponding to two sediment sources.

Parameter	Sediment 1	Sediment 2
Average Grain Size (micro-m)	150	35
Grain size Stand. Dev. (micro-m)	50	30
Porosity (fraction)	0.15	0.15

Table 6-2. Summary of properties for two sediment sources in an assorted, composite network.

The sediment fraction, defined as the fraction of high quality sediment (1) within the total sediment amount, is modified from 0 to 1. For a sediment fraction of 0, the sediment 2 occupies the entire model. Conversely, a pore network model with a sediment fraction of 1 considers only the properties of the sediment 1. Static properties and multiphase flow functions for the composite pore network model with different sediment fraction are evaluated. Figure 6-12 shows results for two assorted composite pore network models with different sediment fractions.

The absolute permeability of the pore network model for different sediment fractions exhibits a sharp transition between the individual permeabilities of the two sediments. The permeability values are particularly influenced by the low quality sediment (as expected) (see Figure 6-13). The multiphase flow properties of the composite pore network model at different values of sediment fraction fall between the bounds for the pure sediments. The residual oil saturation decreases with increasing values of sediment fraction due to the drop in the number of isolated large pore bodies where the oil is easily bypassed and trapped during the imbibition. Thus, as the number of large pore bodies increase further, the connectivity between these large pore bodies is gradually improved, creating paths to displace the oil blobs to the outlet of the medium and consequently reducing the residual saturation at the end of imbibition.

At low values of sediment fraction, the oil relative permeability end point shows an initial reduction. The behavior of the water relative permeability end point with the sediment fraction is consistent with the behavior of the oil residual saturation (see Figure 6-13 and Figure 6-14). The water relative permeability end point normally decreases as the oil residual saturation increases. The initial reduction in the oil relative permeability end point with the sediment fraction is caused by the presence of isolated large pore bodies that contribute to the oil saturation but not to the oil conductivity.

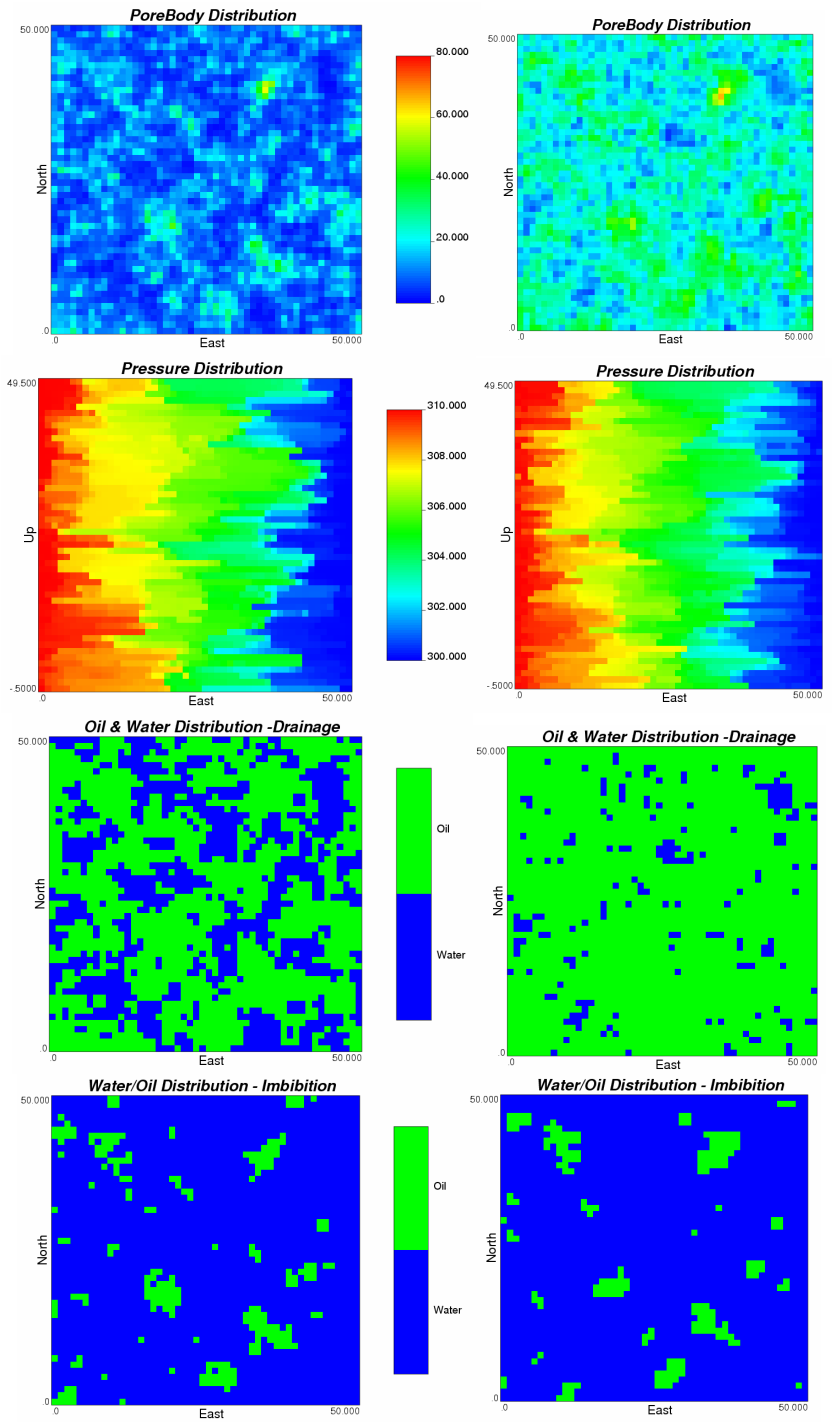


Figure 6-12. Spatial distributions for pore bodies, single phase pressures and phases at the end of primary drainage and imbibition, for two assorted, composite pore network models with sediment fractions of 0.2 (left) and 0.8 (right).

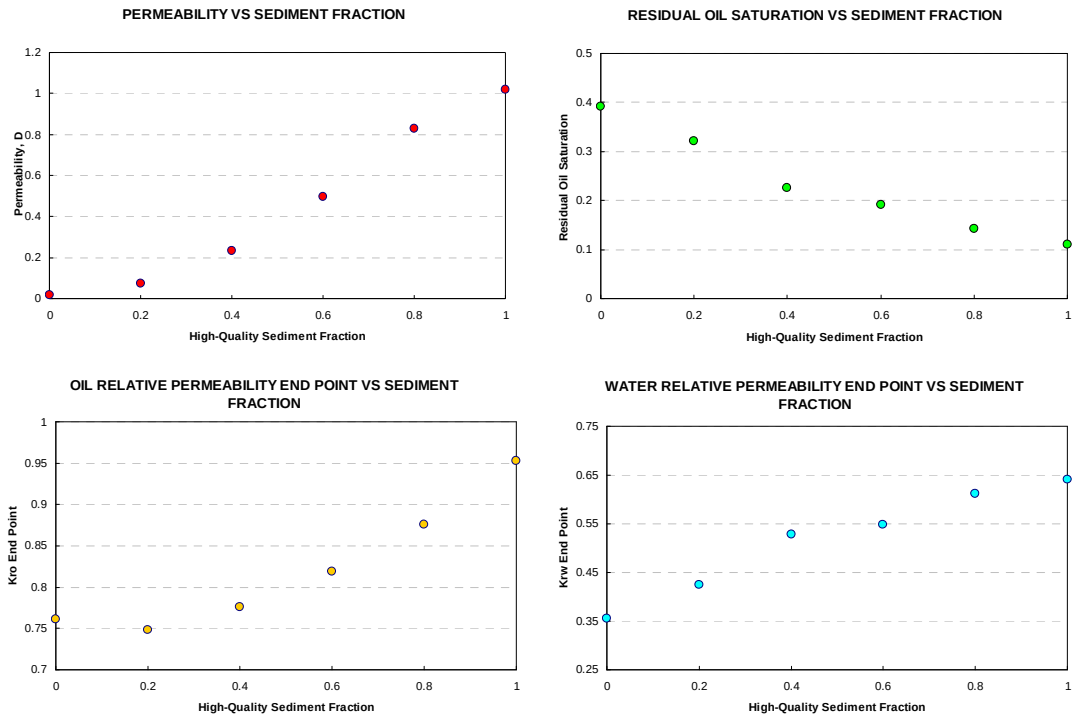


Figure 6-13. Results of the sensitivity study for the influence of relative sediment fractions in the static and multiphase flow properties of an assorted, composite pore network model with two sediment sources.

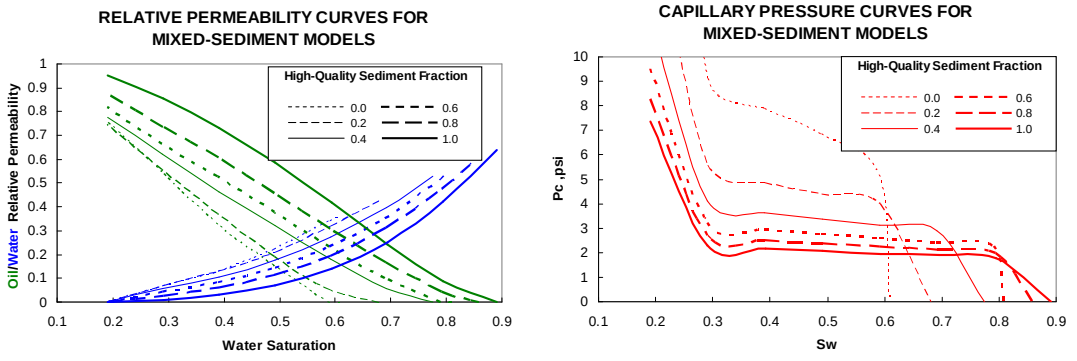


Figure 6-14. Effect of relative sediment fractions on relative permeability and capillary pressure curves of an assorted, composite pore network model with two sediment sources.

6.7 FLOW BASED SCALING OF FLOW FUNCTIONS

Multiphase flow functions calculated with a pore network model are representative at the scale of one inch or less. However, multiphase functions required as input in a flow simulator should be representative at the scale of the average grid size of the simulation model, which is normally of the order of dozens to hundreds of feet. The scale up of petrophysical properties, including multiphase flow functions, is a complicated problem that has to take into consideration, the geological heterogeneity at different scales. In this work, a flow-based approach is proposed to scale up the results of the pore network models to obtain multiphase flow functions representative for flow simulation models.

This approach utilize a small simulation model with a total size equivalent to an expected flow simulation grid cell, and composed of small grid cells with size corresponding to individual pore network models. The method basically reproduces in a flow simulator the steady state laboratory technique for determining relative permeabilities. Static properties and multiphase flow functions calculated from pore network models are spatially distributed through the grid of the simulation model according to the expected spatial distribution of rock types. Thus, composite models of uniformly mixed sediments, layered systems or any other spatially correlated configuration can be represented in this technique. Then, oil and water are injected

simultaneously at different relative rates into one side of the model and produced at the opposite side until steady state is achieved (stabilization of production rates, fluid saturations and pressure drop across the model). Once steady state is reached at a particular fixed relative rate, the pressure drop across the model, the average difference between the phase pressures and the water saturation are acquired. The average difference of the phase pressures at different values of water saturation determine the effective capillary pressure curve for the simulation model. At each fixed relative rate, the pressure drop across the model for each phase is used to determine the relative permeabilities for the correspondent water saturation. The relative permeabilities are calculated using the following equations:

$$k_{ro} = \frac{q_o \mu_o L}{k A \Delta P_o} \qquad k_{rw} = \frac{q_w \mu_w L}{k A \Delta P_w} \tag{6.1}$$

where the oil and water phases are indicated by the subscripts o and w, respectively. k_r is the phase relative permeability, q [cc/s] is the phase rate in, μ [cp] is the phase viscosity, L [cm] is the length of the model in the flow direction, k [D] is the absolute permeability, A [cm^2] is the cross sectional area perpendicular to the flow direction and ΔP [atm] is the phase pressure drop across the model. Injection flow rates for both phases should be as low as possible considering that we are assuming capillary controlled displacements. The absolute permeability is estimated with the same procedure but injecting only water in a simulation model fully saturated with water. Additionally, by modifying the flow orientation, the anisotropy values of the absolute permeability can be determined.

The examples in Figure 6-15 illustrate the results of this flow based method to upscale flow functions for simulation models with different configurations of spatially correlated petrophysical properties. The examples include a uniform model, to prove the reliability of the upscaling method in recovering single rock type flow functions; a layer model; a spatially correlated model, and finally; a random model. Petrophysical properties corresponding to three rock types are used in these models.

A few observations can be obtained from the results. In mixed models, the end point of relative permeability to water in the upscaled curve is always lower than or closer to the lowest end point to water in the single rock-type curves. The upscaled end point of relative permeability to oil and the residual oil saturation always fall in the range defined by the values from single rock-type curves (normal behavior for absolute permeability and porosity). However, the upscaled relative permeability end point to oil falls closer to the best rock-type end point, while the upscaled residual oil saturation is closer to the worst rock-type residual oil saturation. The increment in the oil residual saturation justifies the reduction in the water relative permeability end point in mixed models. These observations apply for mixed models with similar proportions of rock types. Otherwise they would apply only to the most recurrent rock types in the mixture.

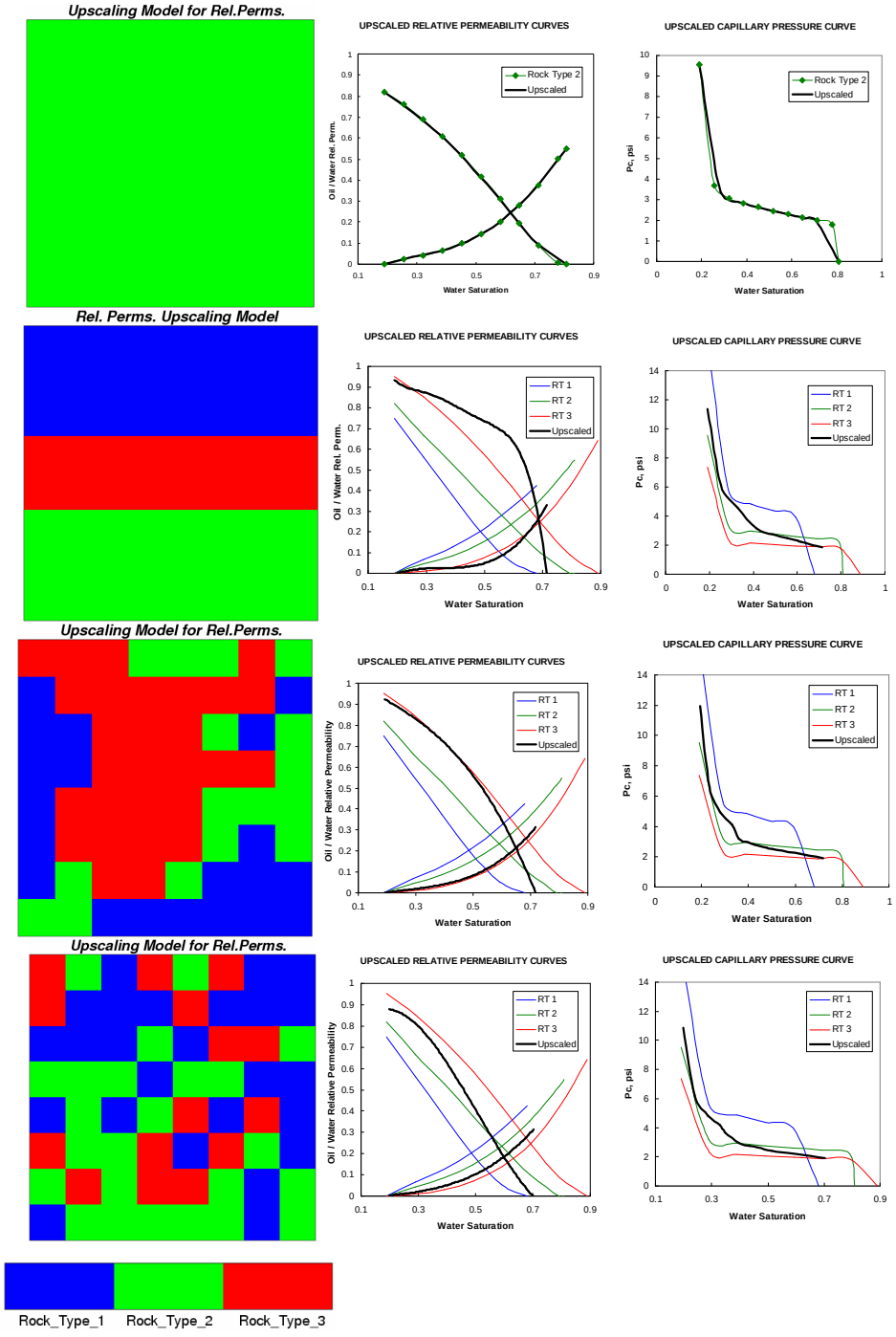


Figure 6-15. Results of the steady state method implemented on Eclipse® flow simulator to upscale multiphase flow functions from single rock types to mixing models with different spatial distributions of rock types.

Due to capillary forces, water is associated to rock types with smaller throats and pore bodies, while oil is associated to large pore bodies and throats. Consequently, as the water saturation increases, water occupies first the low quality rocks while the high quality rocks remain mostly filled with oil. Since the absolute permeability of a flow models is highly influenced by the connected path of high quality rocks, the effective permeability to oil remains high as long as the high connectivity path remains saturated with oil. This explains the change in curvature of the relative permeability curves in models that exhibit spatial correlation of rock types (see layer and correlated models in Figure 6-15). These preferential paths caused by capillary forces also promote water bypasses and oil traps, increasing the residual oil saturation. In turn, increments in the residual oil saturation contribute to reductions in the relative permeability end point to water. An example of an Eclipse simulation file used to upscale the flow functions is presented in Appendix E.

7 HISTORY MATCHING BY PERTURBING NETWORK CHARACTERISTICS

After the evaluation and validation of the history matching algorithm (Appendix C) and the pore network simulator (Appendix A), the next challenge was the development of a combined approach for history matching considering the calibration of a geological model with a distribution of multiphase flow functions consistent with the static properties.

During the history matching process the spatial distribution of rock types is modeled using sequential indicator simulation and calibrated with the production history using the probability perturbation approach. The sequential indicator simulator was modified to generate the files required by the flow simulator, including the distributions of porosity, permeability and multiphase flow function number (saturation function number) based on the distribution of rock types.

7.1 REFERENCE MODEL

For the reference reservoir model in the first application case, a reservoir with four wells (two producers and two injectors) was chosen. The reservoir is assumed to

be a typical fluvial system with channels (see Figure 7-1). The model exhibits a progressive decrease in permeability with depth. This reference model was not generated by and can not be exactly reproduced with a variogram-based heterogeneity model. A regular 100x100x5 grid (50000 blocks) with block dimensions of 30x30x15 is used to represent the reservoir, covering an area of 200 acres. Production history for a period of 1020 days was generated considering initial primary recovery for 210 days using two producers. This is followed by a secondary recovery period with water injection in two additional wells.

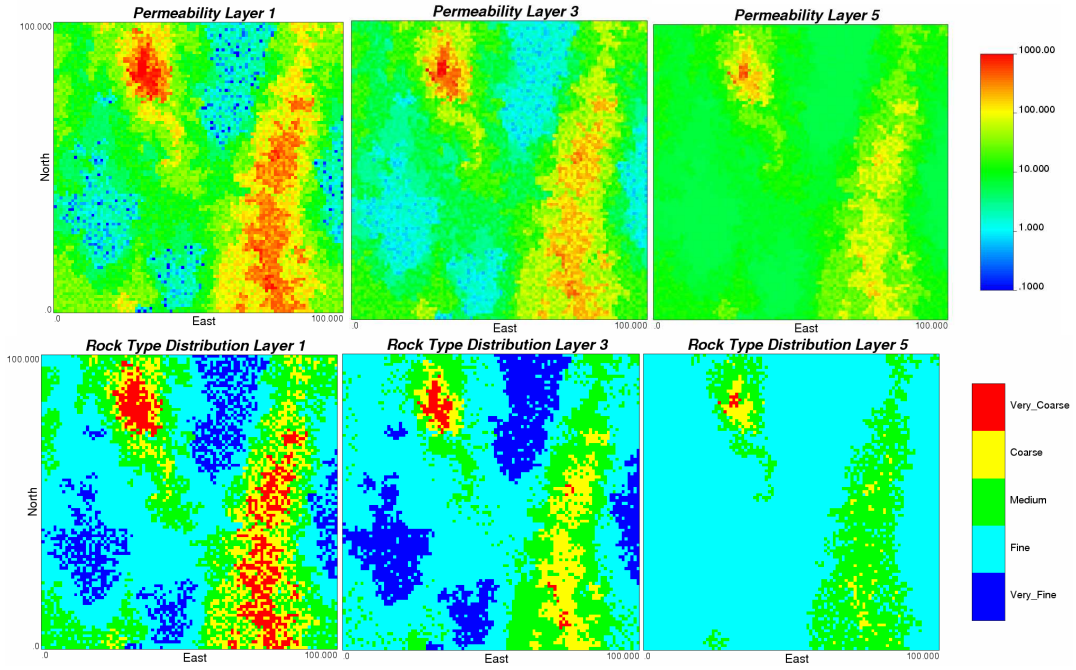


Figure 7-1. Distributions of permeability (top) and rock type (bottom) for layers 1, 3 and 5 of the reference model in the case study.

Producers and injectors are constrained with a bottom-hole pressure of 1800 psi. and an injection rate of 3000 Bbl/day per well, respectively. The initial reservoir pressure is 5000 psi. The flow simulations for this case were generated using a black oil model. The production responses considered for history matching include field pressure, well oil producing rates and water cuts. All these properties were equally weighted in the objective function of the optimization problem.

7.2 MULTIPHASE FLOW LIBRARY

The geological model is based on the spatial distribution of five different rock types corresponding to channel fill, mid bar, levee and transition lithologies (see Figure 7-1). The main difference between the rock types is the average grain size which ranges from very fine (20 microns) to very coarse (180 mm). However, differences in sorting and porosity are also important (see Table 7-1). A pore network model was generated for each of these rock types (see Figure 7-2), and pore network displacement simulations were run to generate the individual multiphase flow functions (relative permeability and capillary pressure curves) and static properties (absolute permeability and porosity). The length of the pore network models is 0.5 inches and includes 125000 pore bodies. The library or collection of multiphase flow functions corresponding to the five rock types were use to estimate upscaled functions

to be loaded into the simulation model. The spatial correlation of basic rock types was considered in the models used for the upscaling procedure. The flow functions for the five rock types are shown in Figure 7-3.

Property	Rock Type				
	Very Fine	Fine	Medium	Coarse	Very Coarse
Ave. Grain Size µm	18	50	90	125	175
Grain Size Stdev. µm	20	30	60	60	80
Porosity	0.12	0.10	0.12	0.13	0.15
Permeability, mD	1.182	13.879	74.019	159.191	565.24
Ave. Pore Body µm	3.582	9.388	16.758	23.836	33.977

Table 7-1 Input and output parameters for pore network models corresponding to 5 rock types in the synthetic case study.

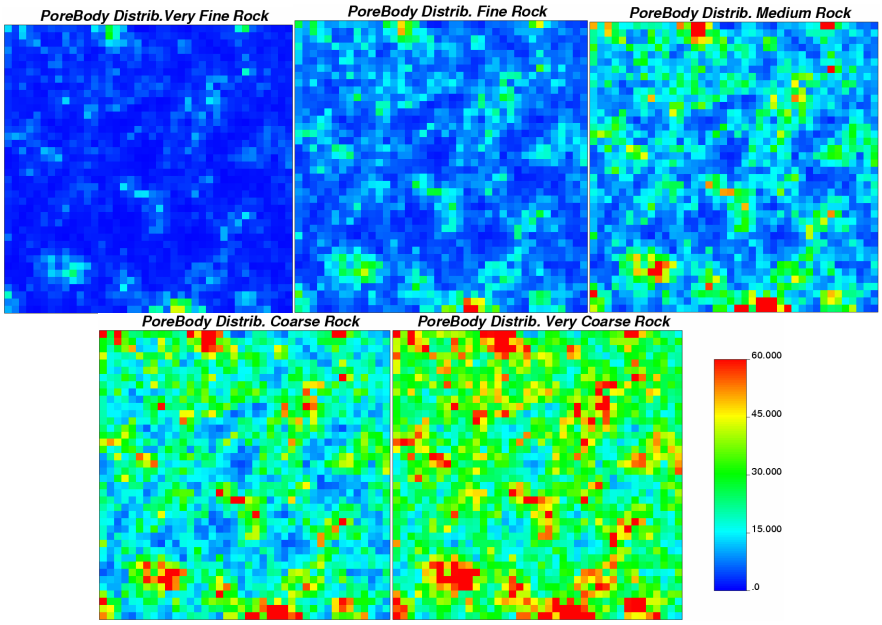


Figure 7-2 Distribution of pore body sizes (microns) for five rock type network models in synthetic case study.

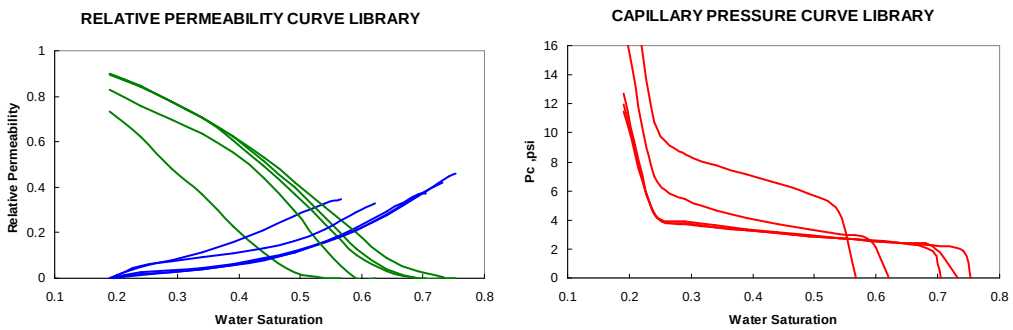


Figure 7-3. Relative permeability and capillary pressure curves for the five rock types obtained by running the pore level simulator.

7.2.1 GEOLOGICALLY CONSISTENT MULTIPHASE FLOW FUNCTIONS

Once the multiphase flow properties are determined by the pore network simulation, they are used as the basis to compute the input functions for the flow simulation. Using the flow functions corresponding to the five basic rock types, the upscaling procedure was applied to determine the multiphase flow functions representative at the scale of the simulation grid blocks for five different rock mixtures. Results are shown in Figure 7-5. Each of the five rock mixtures, shown in Figure 7-4, is characterized by a high proportion of one of the five basic rock types. The spatial distribution of the basic rock types in the rock mixture upscaling models is determined using sequential indicator simulation with correlation lengths consistent with the relative proportions of the rock types. Cubic models with a length of 15 feet were used for upscaling. During the field scale flow simulation, an effective rock type identified by the index representative of the dominant rock type is assigned to each

grid block. The spatial distribution of these indexes is rendered consistent with the expected geologic structure of the reservoir. Consistent with the index assigned to a grid block, upscaled multiphase flow functions and static properties, such as porosity and absolute permeability, are allocated. The history matching process is then based on the perturbation of the grid block indexes.

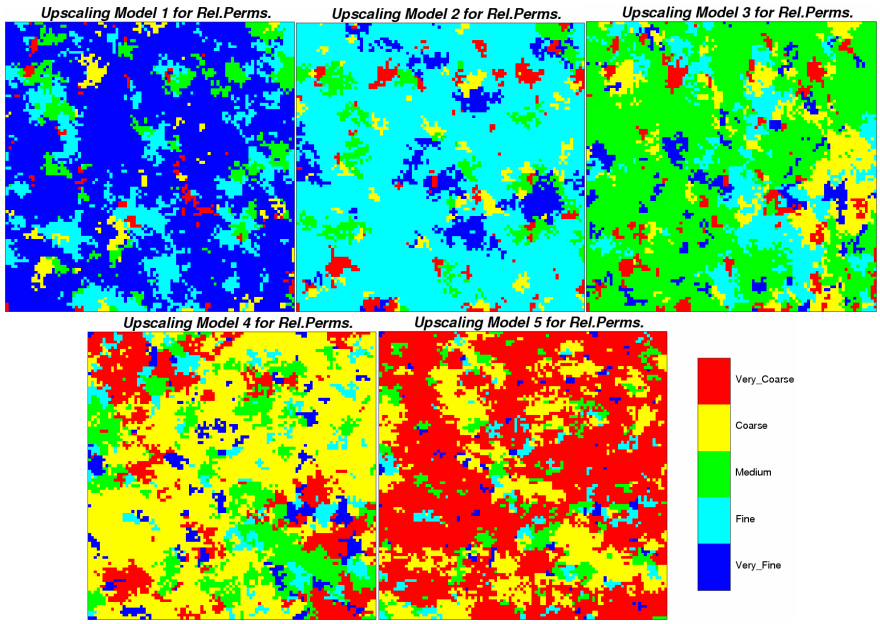


Figure 7-4 Distribution of rock types for five models used to upscale petrophysical properties for the flow simulation model.

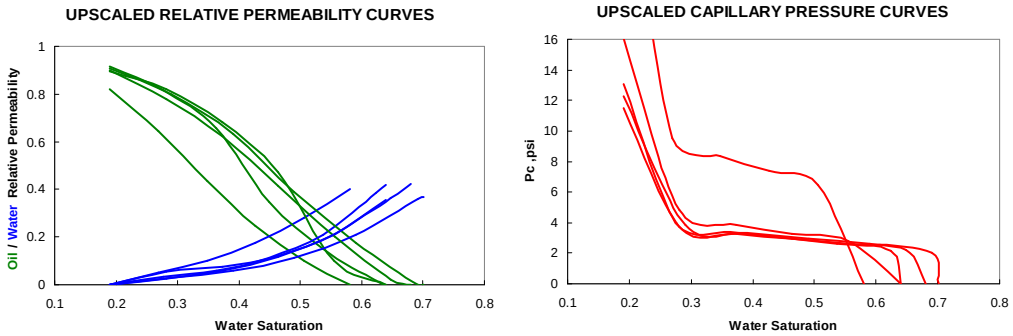


Figure 7-5. Upscaled relative permeability and capillary pressure curves obtained using steady state method on five models considering mixtures of rock types.

To asses the significance of using multiphase flow functions consistent with the geological model, two different flow simulations were run on the reference model. The first simulation considers multiple and geologically consistent multiphase flow functions while the second simulation uses average multiphase flow functions for the entire model (common approach on history matching). Deviations of the production response obtained from both simulations (shown in Figure 7-6 and Figure 7-7) clearly indicate the relevance of using geologically consistent multiphase flow function for this reference model.

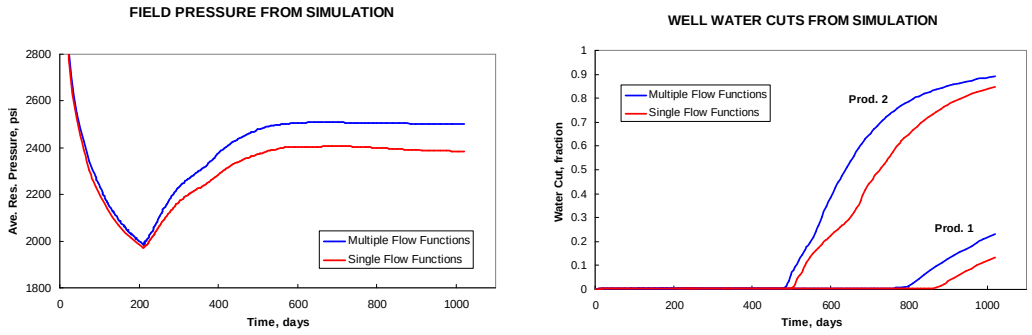


Figure 7-6. Simulation results for the reference model considering two cases, a single set of flow functions (red) and, multiple flow functions consistent with heterogeneity model (blue).

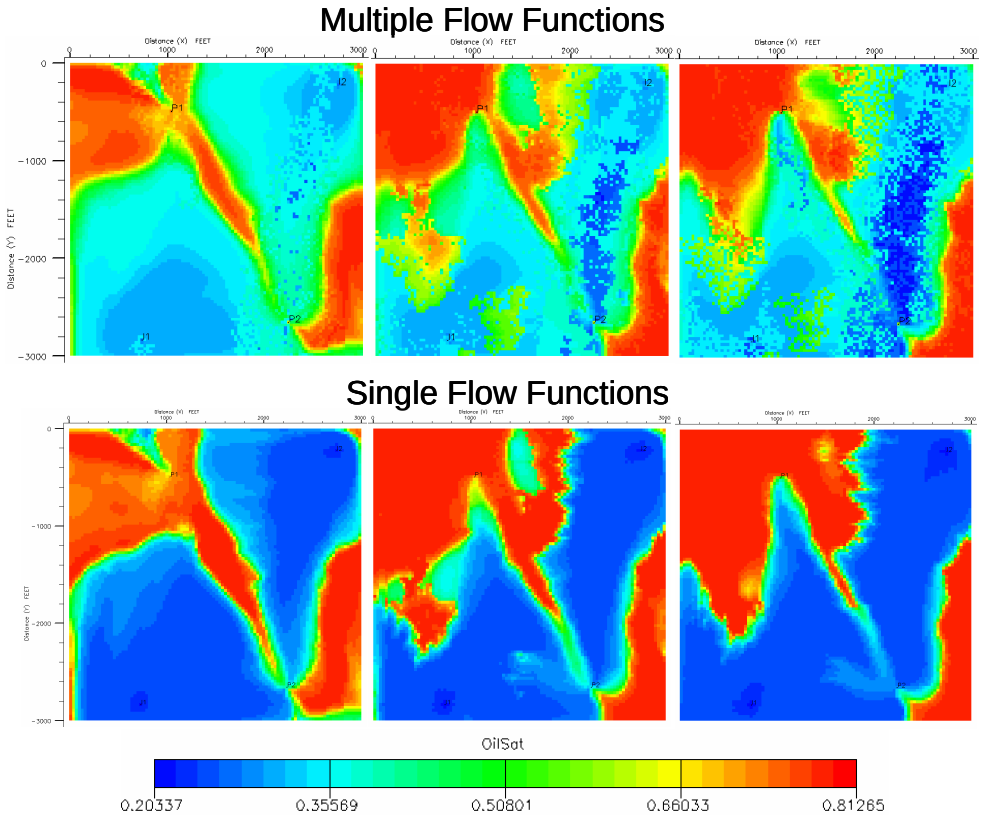


Figure 7-7. Distribution of oil saturation in layers 1, 3 and 5 at the end of the simulation with the reference model considering multiple flow functions (top) and a single set of flow functions (bottom)

7.3 CONDITIONAL DATA AND OBJECTIVE FUNCTION

A total of 45 conditioning values of rock type were sampled from the reference model. These data values were used for generating the variogram model of heterogeneity and are also used as conditional information for the construction of the

initial and subsequent rock type realizations during the process for history matching. The 45 rock type values include 20 at well locations (5 conditioning data per well) and 25 other randomly distributed values through the rest of the model.

The objective function to be minimized during the history matching process is the sum of squared residuals between the simulated production response and the production history (from the reference) over time. Different production variables are considered in the objective function, including field average pressure, and the oil production rate and water cut at both producers. These production variables are normalized by the variance of the production history, so as to give them equal weight towards the objective function.

Sequential indicator simulator is used to generate the initial and subsequent rock type distributions based on the conditional information and variogram models for each rock type. The flow response of the initial realization of rock type distribution has a significant deviation from the production history (initial objective function higher than 100 in Figure 7-9). This is reasonable considering the scarce conditional data (less than 0.1% of the total number of grid nodes) and the fact that the reference model used to generate the production history is a fluvial channel system that can not be exactly reproduced with a variogram model of heterogeneity. Additionally, the reference model exhibits a gradation in permeability values from top to bottom that is not specified in the history matching model. Furthermore, there are

notable discrepancies between the variogram model used in the history matching realizations (estimated from scarce conditioning data) and the “true” one that can be inferred from the reference model.

7.4 HISTORY MATCH RESULTS AND DISCUSSION

During the gradual deformation approach for history matching a double loop Markov Chain procedure (inner-outer loops) is implemented. In the inner loop, the probability perturbation method and a Dekker Brent optimization routine are used to determine the optimal transition from the initial reservoir model to four different proposals generated with the same conditional data and heterogeneity model. In the outer loop, the current optimal model is updated with the best model obtained in the inner loop and four new proposals are generated.

The initial, final and reference realizations during the first history matching case are shown in Figure 7-8. In comparison to the patchy distribution of rock types in the initial model, the final history matched model exhibits the continuity of rock types consistent with the reference model. Some statistics of the spatial distribution of connectivity indices corresponding to the conditioning information, and the reference, initial and final realizations during the history-matching study are presented in Table 7-2. Reproduction of the prior heterogeneity model (conditioning information) in the

initial and final realizations can be observed in this table. Also evident in this table is the difference between the statistics of the reference model and the conditioning information, inferred from very scarce conditioning data. The simulated production response matches fairly close the production history (see Figure 7-10).

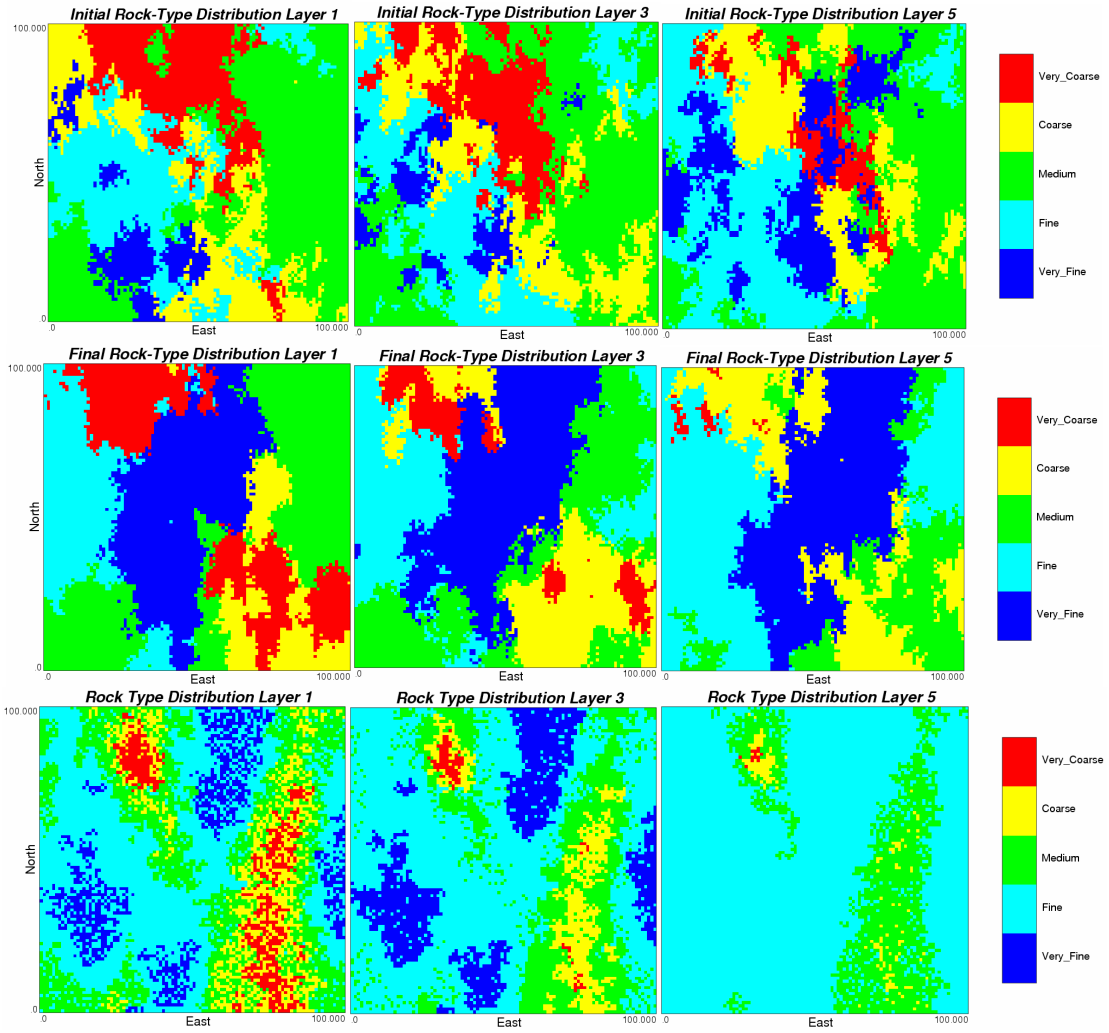


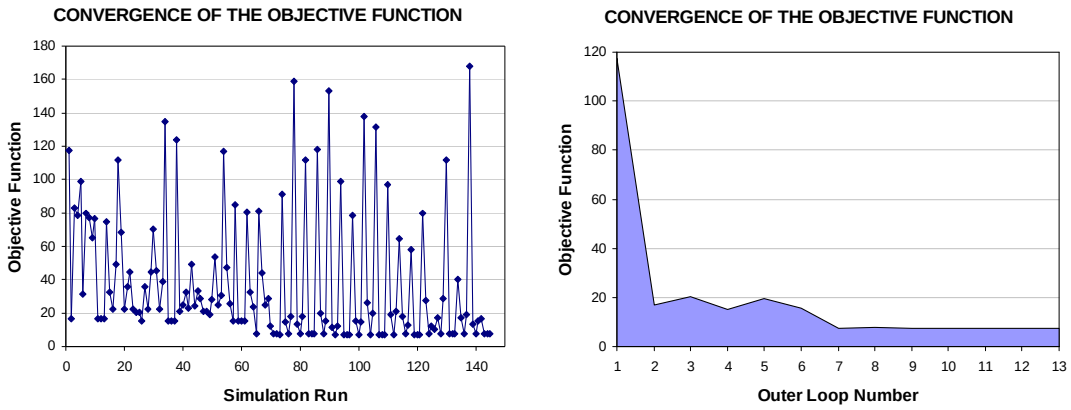
Figure 7-8. Layers 1, 3 and 5 of the initial (top), final (middle) and reference (bottom) models corresponding to the history matching case study.

Property	CI	Model			
		Reference	Cond. Data	Initial	Final
Proportion	1	0.124	0.20	0.159	0.246
	2	0.556	0.26	0.236	0.239
	3	0.227	0.26	0.281	0.278
	4	0.072	0.18	0.198	0.161
	5	0.021	0.10	0.126	0.076
Main Correlation Length ft.	1	990	1200	900	1500
	2	1260	1300	1230	1350
	3	1710	1750	1980	1560
	4	1920	1200	1050	1350
	5	1650	1400	1380	1380
Secondary Correlation Length ft.	1	480	550	510	600
	2	360	950	900	990
	3	390	1050	960	1170
	4	300	700	600	780
	5	300	550	360	780

Table 7-2 Statistics of the spatial distribution of connectivity indices corresponding to the conditioning data, and the reference, initial and final models of the synthetic history-matching case.

Systematic convergence of the objective function is observed through out the gradual deformation process. In the first outer iteration step the objective function drops from 114 to 16. In subsequent steps the convergence is more modest and reaches a plateau after 70 simulations. The final objective function after 75 simulation runs is 7.2. Even though the final value of the objective function is still relatively high, the final model is reasonably good considering that the reference channel model can not be reproduced with the suggested variogram model of heterogeneity. The convergence is also deemed adequate considering that objective function is composed

of five different production variables. The objective function depends on the applied length norm for the mismatch, number of production variables, normalization weights and time steps. Consequently, the absolute value of the objective function alone is not a clear measure for quality of convergence. Convergence can be evaluated by comparing the value of the objective function for the final history-matched model with that corresponding to new stochastic proposals (including the initial model); and the gradual improvement of the objective function with the number of simulation runs until a plateau is reached.



a) b)

Figure 7-9. Convergence characteristics of the history matching algorithm: a) the fluctuation in objective function corresponding to all iteration steps during the process; b) The objective function characteristic of the retained models during the history matching process.

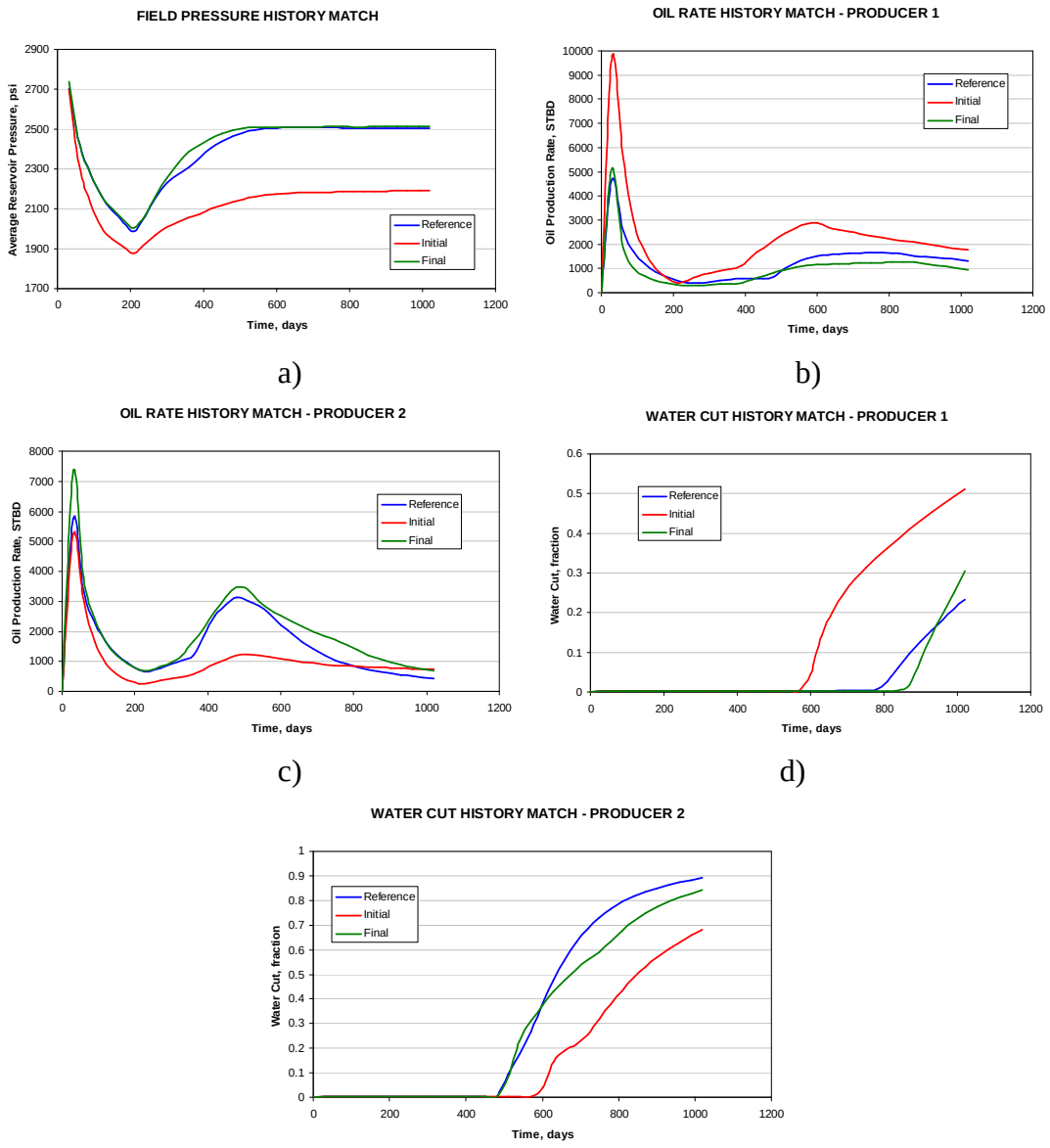


Figure 7-10. History match results: a) Field pressure match; b) Match for oil rate at producer 1; c) Match for oil rate at producer 2; d) Match for water cut at producer 1; e) Match for water cut at producer 2.

Both, the reference and the final models were used to forecast production beyond the history matching period, to confirm the predictive capability of the final

history-matched model (see Figure 7-11). The results indicate that the integration of dynamic data in the reservoir model and the consistent perturbation of all static and flow parameters influences not only the near well region (reflected by the matched oil rate at producers), but also the inter-well region (reflected by the accurate prediction of water cuts).

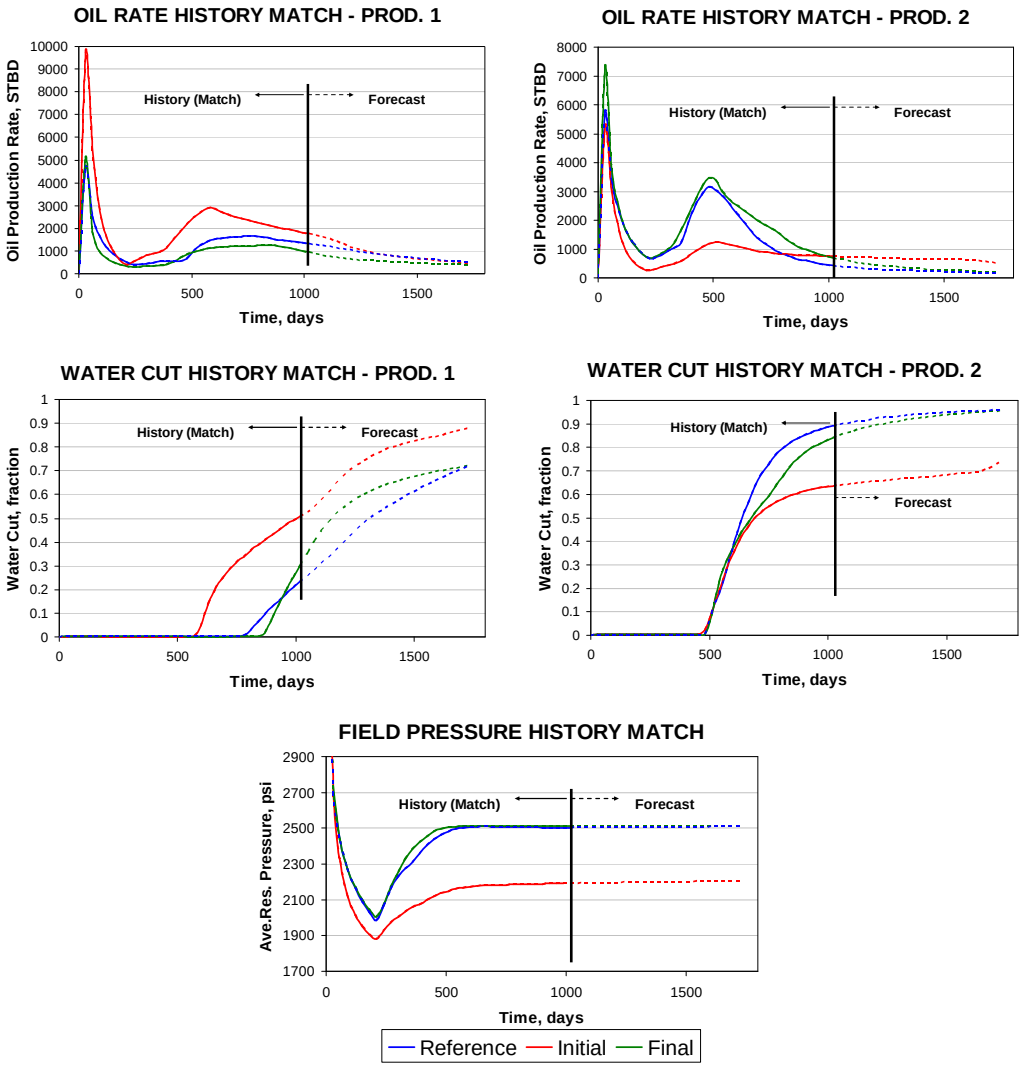


Figure 7-11. Performance prediction based on the initial reservoir model and the final history matched model, compared to the reference.

7.4.1 IMPORTANCE OF GEOLOGICALLY CONSISTENT PERTURBATION OF ROCK PROPERTIES

To assess the importance of using geologically consistent scheme for perturbing the entire suite of multiphase flow functions, a second history matching case was run. In contrast with the first case presented above, a single set of multiphase flow function for the entire reservoir is used in the second case. The flow functions corresponding to the rock type 4 were used in the simulation case with single flow functions. This approach in the second case is a common practice in the industry where a single set of relative permeability curves obtained from core lab studies or using models such as Corey type; are used to represent the rock-fluid interactions in the entire reservoir during history matching. The comparison of the result obtained after history matching using a single set of flow functions against that obtained using spatially varying flow functions, is shown in Figure 7-12 and Figure 7-13. The final model using geologically consistent multiphase flow functions (first case) showed better results in the convergence of the objective function and the forecast of the production beyond the history matching period.

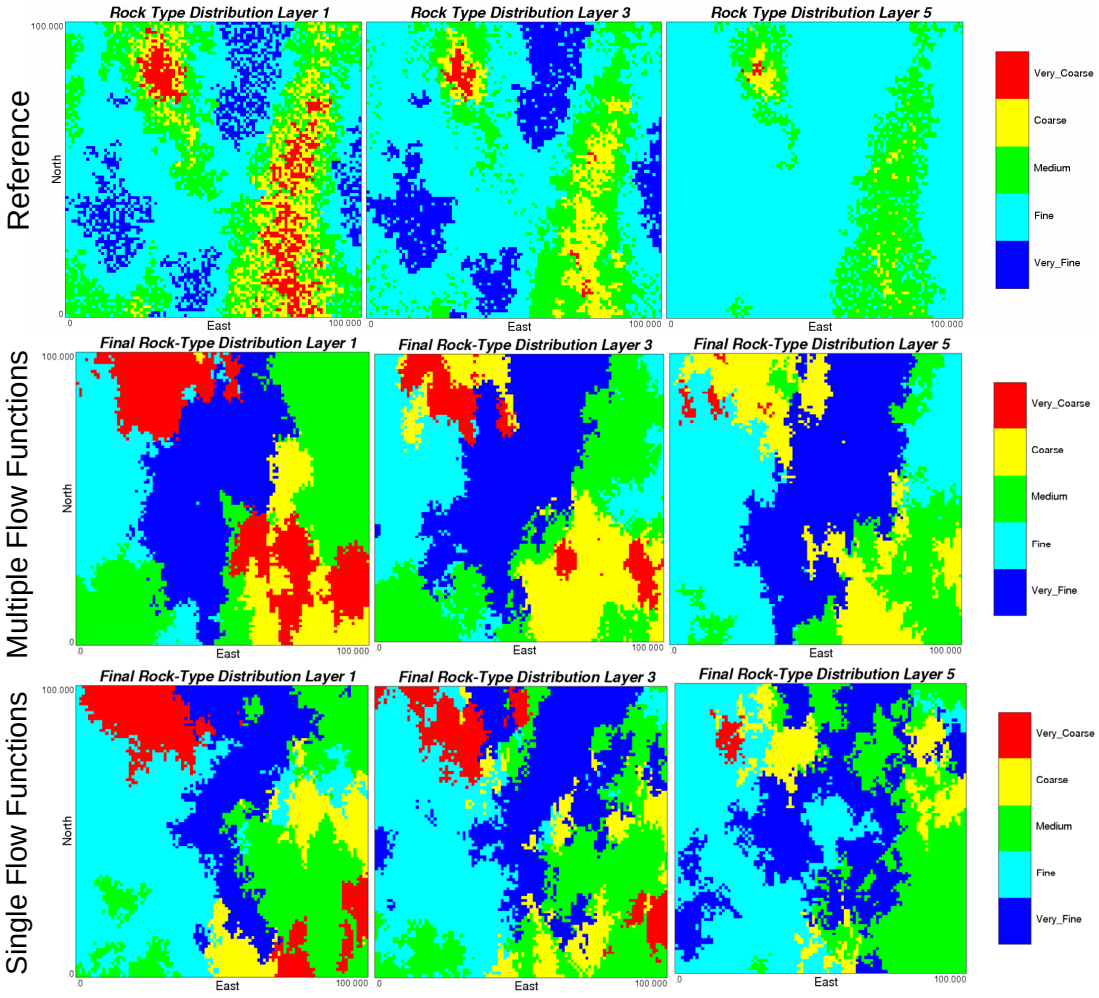


Figure 7-12. Layers 1, 3 and 5 of the reference model (top), and the final models with multiple consistent multiphase flow functions (middle), and single flow functions (bottom), for to the history matching case study.

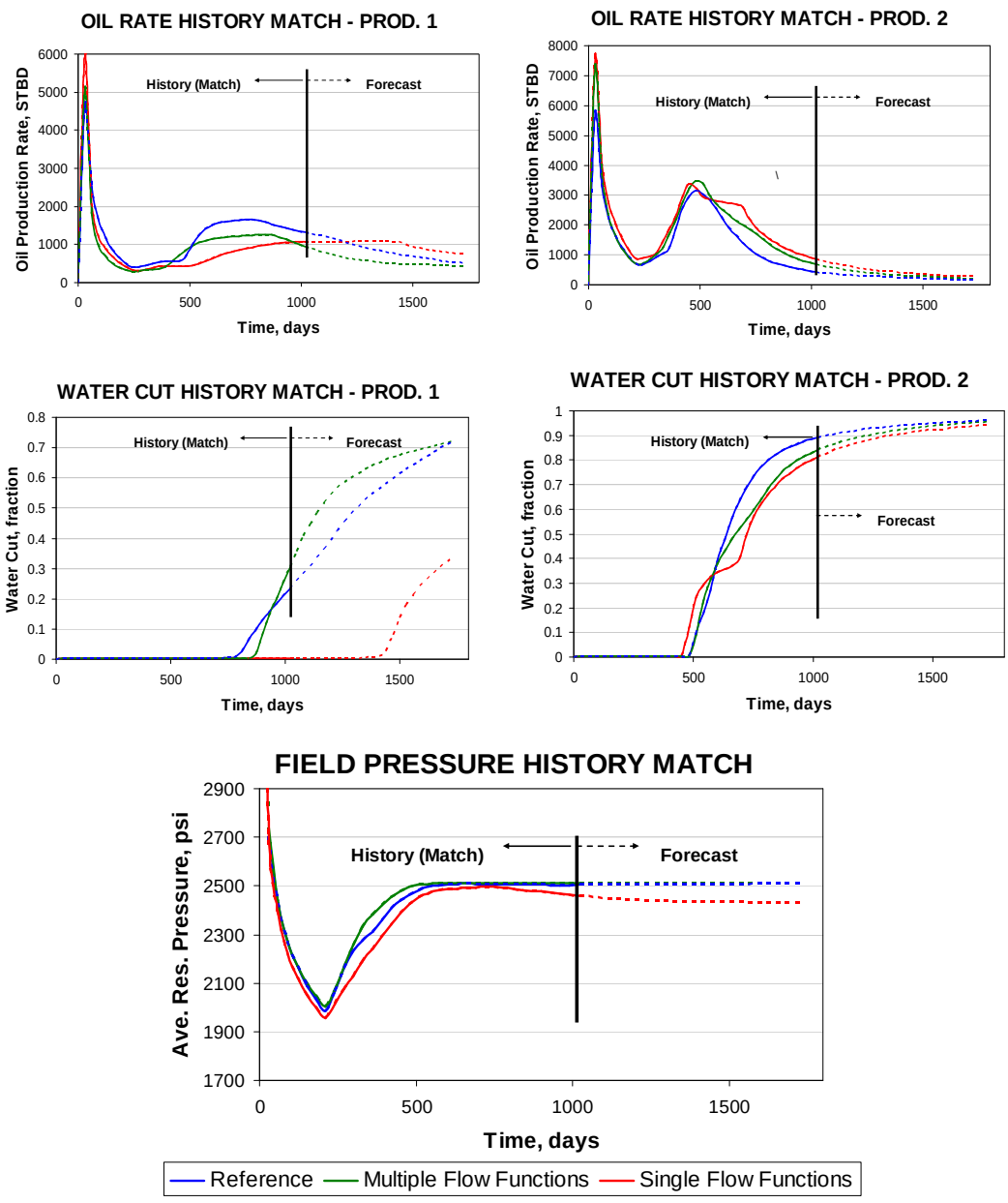


Figure 7-13. Results of history match and forecast production variables for the reference and final realizations of the two history matching cases.

8 FIELD CASE STUDY

In the previous chapter, the proposed approach for history matching by simultaneous calibration of all rock properties in a geologically consistent fashion was applied on a synthetic yet realistic case study. Synthetic models offer the great advantage of providing exhaustive geological data associated with the production history. Thus, an assessment of the proximity between the reference and the history matched models can be performed. However, history matching studies with synthetic models assume that the fluid flow processes are known completely and therefore uncertainty associated with fluid and petrophysical characterization of rocks is overlooked.

In this chapter, the proposed multi-scale approach for history matching is applied to a real field-scale case study. In this study, like most field cases, limited information to accurately depict the geological heterogeneity and rock and fluid characteristics, result in considerably uncertain production forecasts. An assessment of the causes of uncertainty while predicting flow response led us to formulate the proposed procedure to introduce consistency between the geological model and all rock properties including rock-fluid interactions. In this approach, the history matching process is implemented by perturbing the distribution of connectivity

indexes associated with upscaled rock properties. The connectivity index is influenced by the spatial correlation of basic rock types and the corresponding static and flow properties computed from pore level representations.

8.1 RESERVOIR MODEL

The target formation is about 2500 feet thick and is composed of sands deposited during the Eocene age in fluvial-deltaic to shallow-marine environments, as part of transgressive and highstand system tracts. They were deposited in an overall transgressive (retrogradational) setting with lowermost sands having the most on-shore character and the uppermost sequence having the most marine character. The formation was deposited as onlapping wedges across reverse faults. These reverse faults are responsible for major compartmentalization of the formation (Mijares *et al.* 2001).

The compartmentalized formation has multiple areas with several productive sands acting as different pools. The area of interest in this study has an approximate surface area of 1700 acres and has produced over 42 years. The structure is a fault-bounded anticline tilted to the east (see Figure 8-1). The area is controlled by a system of inverse faults in the N-S direction.

This study focuses on producing sands from the top pool of this area with an approximate thickness of 350 ft. These sands form a clastic wedge, thickening to the east and correspond to deltaic deposits from environments ranging from fluvial channels (top) to delta plains (bottom) (Mijares *et al*. 2001). This pool, which from now on will be denoted as the "reservoir," exhibits low dips between 2^{o} and 10^{o} to the east and is bounded on the west by a major fault. This multi-layer reservoir has been produced commingled and appears to be isolated from the other pools based on analysis of pressure response. The driving mechanism in this pool is solution gas while other pools show the influence of an extensive aquifer with moderate activity. The aquifer in the target pool is non-active. The reservoir was subdivided in 5 sands with the top 3 being the best producers. These top 3 sands pinch out towards the top of the structure (East) (see Figure 8-1) The total oil column is estimated in 160 ft with an average net to gross ratio of 0.46.

The field development strategy included 15 years of primary production and over 25 years of waterflooding. The cumulative production for the 13 wells targeting these sands is 26 MMSTB, out of an estimated of 220 MMSTB OOIP. The first two producers were converted to injectors during the waterflood. Primary production depleted the reservoir from an initial pressure of 3200 psi to 1700 psia at a reference depth of 6300 ft. The depletion slowed down during the initial period of waterflood, and later, the pressure stabilized at 1400 psi. The reservoir produces oil with an

average API gravity of 32^{o} and a bubble point pressure of 2300 psi. The water-oil contact, WOC, was located at 6500 ft.

In this reservoir, the complexity of the geological setting due to the transition in depositional environments can not be exactly reproduced with a variogram-based heterogeneity model. A 135x78x10 grid (105300 total and 86714 active blocks) with individual block dimensions of 101 ft x 101 ft and variable thickness is used to represent the reservoir. Primary recovery for the period of 15 years considers the production of 4 wells. This is followed by a 25 years waterflood period with 9 additional producers and two of the initial producers converted to injectors. More details about the simulation model are presented in Table 8-1. Producers and injectors, shown in Figure 8-1c, are constrained with oil production rates and water injection rates according to monthly production records. A black oil simulator (Eclipse® from Schulmberger) is used for this study. The production responses considered for history matching include field pressure, field water cut, and the well water cut from 6 wells with representative production history and strategically distributed through the reservoir (see Figure 8-1c). All these properties were equally weighted in the objective function of the optimization problem.

SIMULATION PROPERTY/DESCRIPTION	VALUE
Simulation Model	Black Oil
Simulation Period, years	43
Grid (Cartesian)	135x78x10
Active Grid blocks	86714
Grid block dimensions, ft^3	101x101xVariable
Porosity	0.08 – 0.25
Kx = Ky (Mean – Std Dev), md	5 - 700
Kz/Kx	0.1
Saturation Pressure, psi	2295
Water-Oil Contact, ft	6530
Reference Depth, ft	6300
Initial Pressure @ 6300 ft, psi	3200
Residual Water Saturation	0.21
Residual Oil Saturation	0.25 – 0.30
Oil Relative Permeability Endpoint	0.78 – 0.90
Water Relative Permeability Endpoint	0.25 – 0.28
Oil Gravity (API)	32
Water Injectors	2
Injection Control	Rate History
Injection – BHP upper limit, psi	6000
Oil Producers	13
Production Control	Rate History
Production – BHP lower limit, psi	400

Table 8-1. Description of the simulation for the field case application of the proposed history matching approach.

The initial reservoir model was built considering information from well log interpretations for 42 wells and core lab analysis for samples from 5 wells. The 42 wells, shown in Figure 8-1c, are located in the reservoir area, but most of them produce from lower pools. Based on log interpretations, the reservoir top and total thickness for each of the five producing sands at the 42 well locations were used to construct the reservoir structure using ordinary kriging. The producing sands were

divided to obtain 10 layers with similar thickness. The three middle sands were splitted into 2 layers each, while the bottom sand was divided into 3 layers.

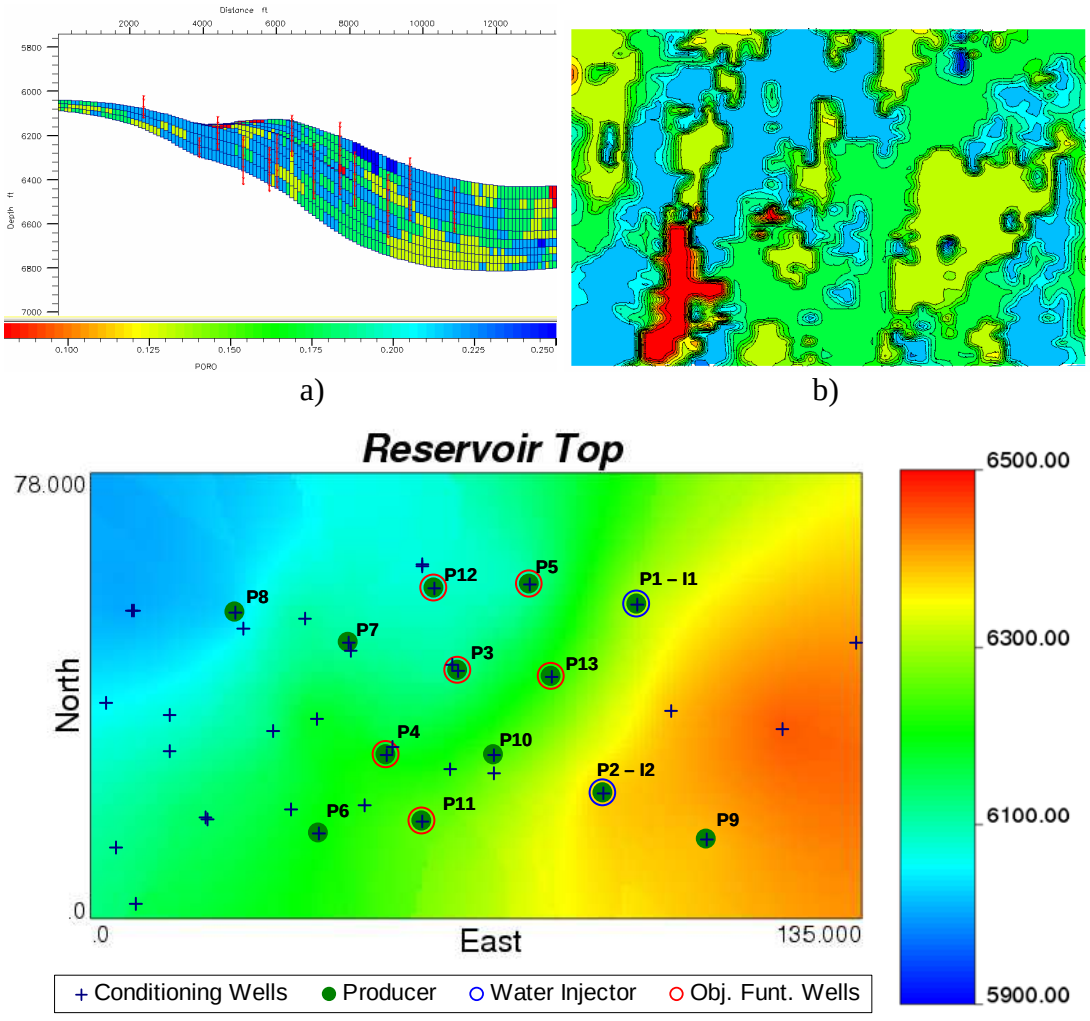


Figure 8-1. Reservoir model: a) Cross section through the initial porosity model; b) aerial map of porosity on layer 3; c) Reservoir top and location of the production, injection and wells that were used for conditioning. The figure also shows the location of those wells that were used in the objective function calculations.

According to well log interpretations, net-to-gross values show clear directional trends in the reservoir. The distribution of net-to-gross values was estimated based on log data for each of the sands at the 42 well locations, using sequential Gaussian simulation. An initial value for the water-oil contact was estimated based on well log values of water saturation from the first two producers. Later, the distribution of net-to-gross values and the WOC depth were further refined to match the estimated original oil in place (220 MMSTB) and the initial water cut reported in production records. During the history matching process, the reservoir structure, the WOC and the distribution of net to gross values are considered deterministic. On the other hand, the distribution of petrophysical properties including absolute permeability, porosity and the distribution of consistent multiphase flow functions will be calibrated through the proposed probabilistic approach for history matching. Details about the distribution of petrophysical properties through the reservoir model will be presented on the following sections.

8.2 GEOLOGICALLY CONSISTENT MULTIPHASE FLOW FUNCTIONS

According to core data analysis, porosity and permeability values from samples exhibit a bimodal distribution representing a mixture of two sediment sources. High quality sediments (large average grain size and good sorting) with higher porosity and permeability values are mixed with low quality sediments (small

average grain size and relatively poor sorting). The first sediments are mostly associated with clean sands in high-energy depositional environments such as fluvial channels, distributary channels and distributary mouth bars. On the other hand, the low quality sediments are linked to low-energy deposits such as delta plain interdistributary and crevasse facies. The spatial variability in the relative proportions of these sediments is the main cause of heterogeneity in this reservoir. The proposed geological model is based on the spatial distribution of rock types that exhibit different relative proportions of the two basic sediments. The fraction of the poor quality sediment varies between 0 and 40 percent of the total sediments, according to core data. Five different rock types were deemed appropriate to characterize the variation of the low-quality sediment fraction within this range.

Network models representing the pore structure of rocks containing mixtures with different relative proportions of the basic sediments were used to estimate the fundamental rock properties (including static and flow characteristics). Thus, five cubic network models with a length of 0.8 inches and different volume fractions of low quality sediments (0, 10, 20, 30 and 40 percent) were generated. The basic properties of these network models, including the average grain size, sorting and average porosity, were calibrated to reproduce results from basic core analysis, such as porosity-permeability correlation, residual saturations and relative permeability end points. The high quality sediment has a grain size distribution characterized by an average of 80 microns and a standard deviation of 25 microns. On the other hand, the

grain size distribution of the poor quality sediment has an average of 20 microns and a standard deviation of 20 microns. Table 8-2 shows other input and output parameters for these network models. Pore network models and the corresponding petrophysical properties are shown in Figure 8-2 and Figure 8-4, respectively. The five rock types (RTs) could be associated with characteristic lithologies from deltaic deposits. For example the rock type 1 (RT1) with a 40% fraction of low quality sediments could be associated to interdistributary-crevasse facies. Similarly, delta front slope, distributary channel, fluvial channel and distributary mouth bar facies could be associated with rock types 2 through 5 respectively, considering a decreasing fraction of low quality sediments. Gas relative permeabilities can not be estimated from the implemented pore network simulator. Consequently, the same gas relative permeability, borrowed from a sample core analysis for a close by reservoir with similar geology and fluid properties, was used for all the rock types.

Property	Rock Type				
	1	2	3	4	5
Poor Sediment Fraction	0.0	0.1	0.2	0.3	0.4
Porosity	0.05	0.10	0.15	0.20	0.25
Permeability, mD	5.42	34.67	126.42	340.18	679.95
Residual Oil Saturation	0.32	0.26	0.22	0.19	0.17
Ave. Pore Body, μm	6.05	8.54	10.99	13.45	15.92
Ave. Connectivity	3.11	3.19	3.28	3.38	3.49

Table 8-2 Input and output parameters for pore network models corresponding to 5 rock types in the field case study. These models consider a mixture of two sediment sources in different relative proportions.

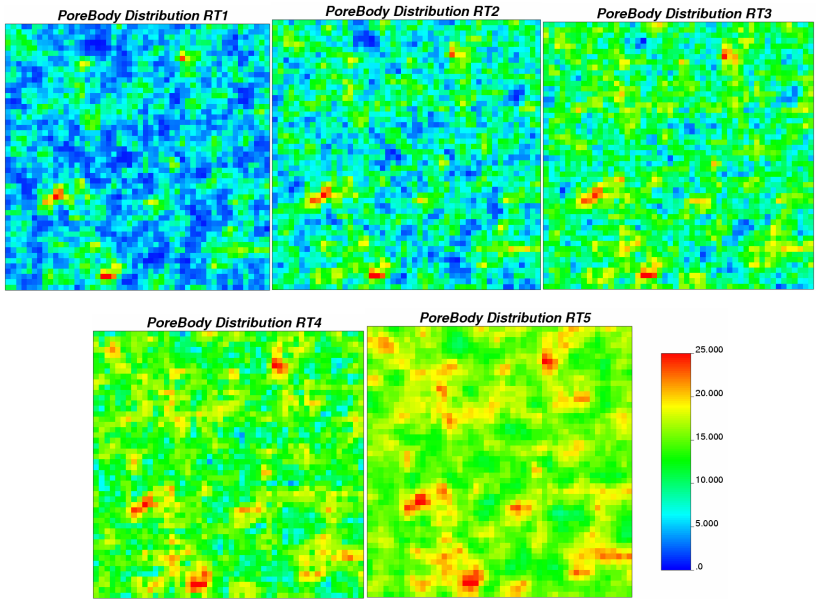


Figure 8-2. Distribution of pore body sizes (microns) for five rock type network models.

The fundamental rock properties computed using network models were used as the basis for obtaining upscaled rock properties representative of grid block volumes. For the upscaling procedure with the steady state method, the spatial distribution of basic rock types was assumed based on appropriate spatial correlation and anisotropy models corresponding to deltaic facies associated with the rock types. For instance, high quality rock types, such as fluvial channel and distributary mouth bar facies, tend to exhibit high anisotropy while low quality rock types, such as interdistributary-crevasse facies, tend to be more isotropic at the scale of simulation grid blocks. Thus, five different models considering transitional states between high and low quality rocks at grid block scale were constructed using the five basic rock types and were used to compute the upscaled rock properties (see Figure 8-3). The

dimensions of the mixing models are defined by a square area with a side length of 50 ft and a thickness of 25 ft. Connectivity indexes were assigned to each of these upscaling models. The field scale distribution of connectivity indexes will be modeled and perturbed during the history matching process. An example of an Eclipse simulation file used to upscale the flow functions is presented in Appendix E.

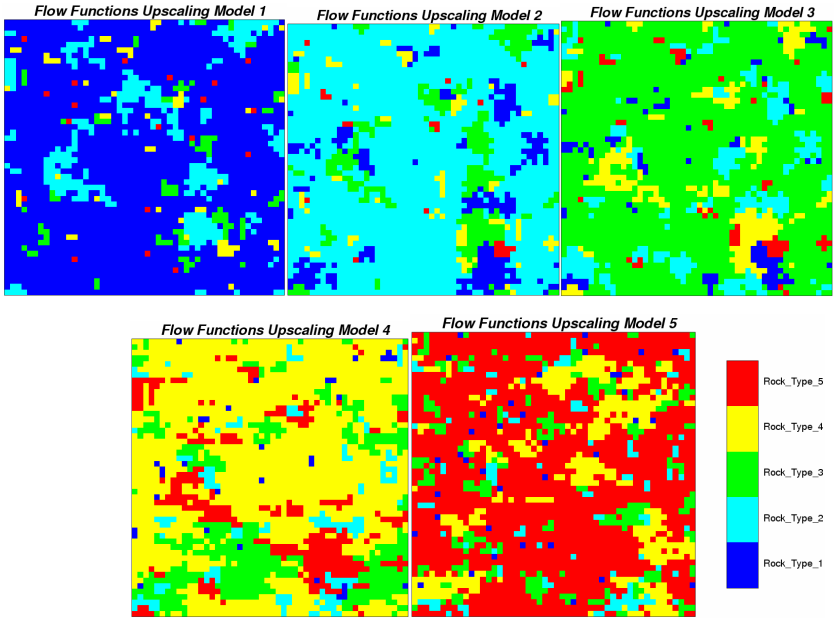


Figure 8-3. Distribution of rock types for five transition models used to upscale petrophysical properties for the flow simulation model. Each color code in these figures corresponds to a rock type with pore body size distribution shown in Figure 8-2.

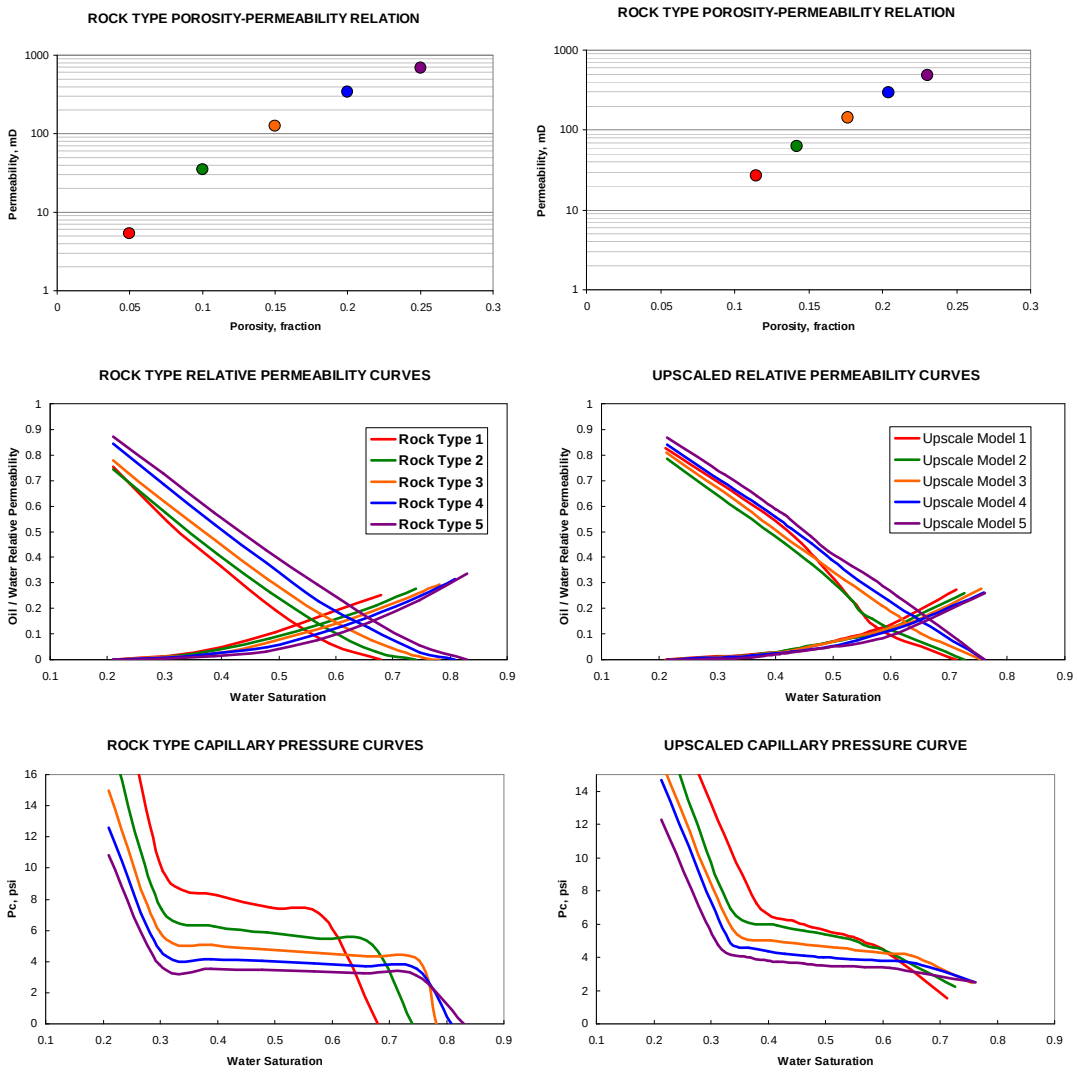


Figure 8-4. Difference in the porosity-permeability relation (top), relative permeability (middle) and capillary pressure (bottom) curves corresponding to rock types (left) and the upscaled functions (right), that consider mixtures of rock types.

To asses the significance of using multiphase flow functions consistent with the geological model, two different flow simulations were run on the same initial realization of the reservoir model. The first simulation considers a spatial distribution

of the five connectivity indexes. Thus this simulation considers multiple and geologically consistent multiphase flow functions while the second simulation uses average multiphase flow functions for the entire model (common approach on history matching). Deviations of the production response obtained from both simulations (shown in Figure 8-5) indicate the sensitivity of the reservoir model response to the spatial distribution of multiphase flow functions.

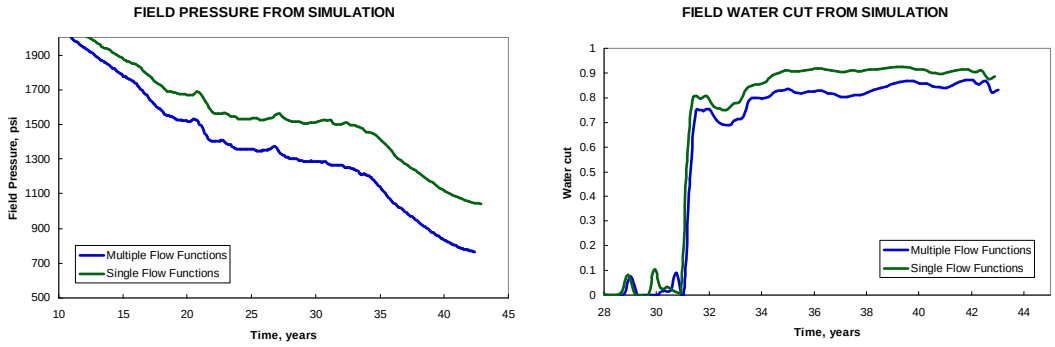


Figure 8-5. Simulation results for the same initial model considering two cases, a single set of flow functions (green) and, multiple flow functions consistent with heterogeneity model (blue).

8.3 CONDITIONING DATA AND OBJECTIVE FUNCTION

A total of 139 conditioning values of connectivity index were estimated based on petrophysical properties from core analysis and well log interpretation. Connectivity indexes were assigned to individual grid blocks based on average values of permeability and porosity, wherever available. Estimated data ranges for these

petrophysical properties corresponding to individual connectivity index values are shown in Table 8-3. These values were used for computing the field-scale variogram model of heterogeneity and are also used as conditional information for the initial and subsequent connectivity index realizations during the process for history matching.

Connectivity Index	Permeability Range	Porosity Range
1	< 10 mD	<10%
2	10 – 50 mD	10 – 15 %
3	50 – 150 mD	15 – 18 %
4	150 – 400 mD	18 – 22 %
5	> 400 mD	> 22%

Table 8-3. Data ranges for permeability and porosity corresponding to each connectivity index for the field case study.

The 139 connectivity index values correspond to 42 well locations including the 13 wells that produce from the target reservoir. Conditioning and production wells are shown in Figure 8-1). The additional wells were drilled through the reservoir but produce only from lower pools. Most of these additional wells are located in the top of the structure (West) where only the two lower and worst sands are present. Consequently, most of the conditioning data from the additional wells is located in the lower sands. The best sands (top 3) pinch out towards the top of the structure (see Figure 8-1). Other useful conditional information also available at the 42 well locations was used to build the reservoir model, including the top of the reservoir and; thickness, water saturation and net to gross for each sand.

The objective function or mismatch between the simulated production response and the production history (from field records) is calculated as the sum of squared residuals and aggregated over time. Field production history is presented in Figure 8-6. Again, different production variables are considered in the objective function, including average field pressure, field water cut, and well water cut from 6 different wells. The 6 wells were chosen because of their representative production history and strategic location in the reservoir (see Figure 8-1). These production variables are normalized by the variance of the production history, to give them equal weight towards the objective function.

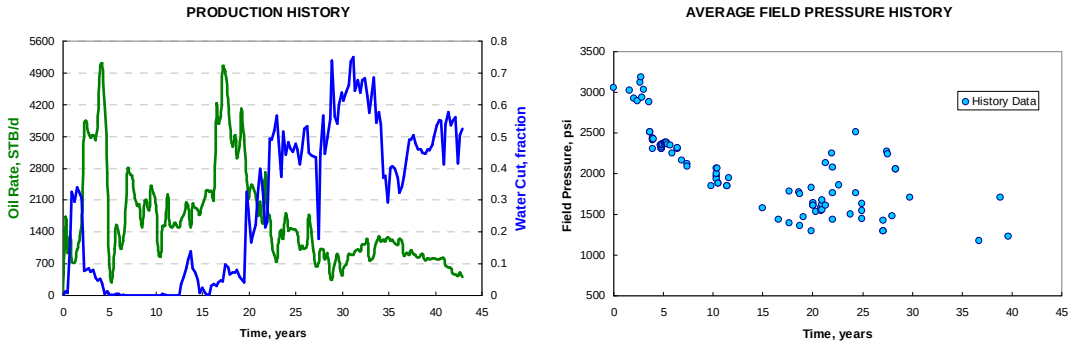


Figure 8-6. Field production history data, including oil production rate, water cut and average pressure.

Sequential indicator simulator is used to generate the initial and subsequent connectivity index distributions based on the conditioning information and variogram models. The flow response of the initial realization has a significant deviation from the production history (initial objective function higher than 600 in Figure 8-7). These

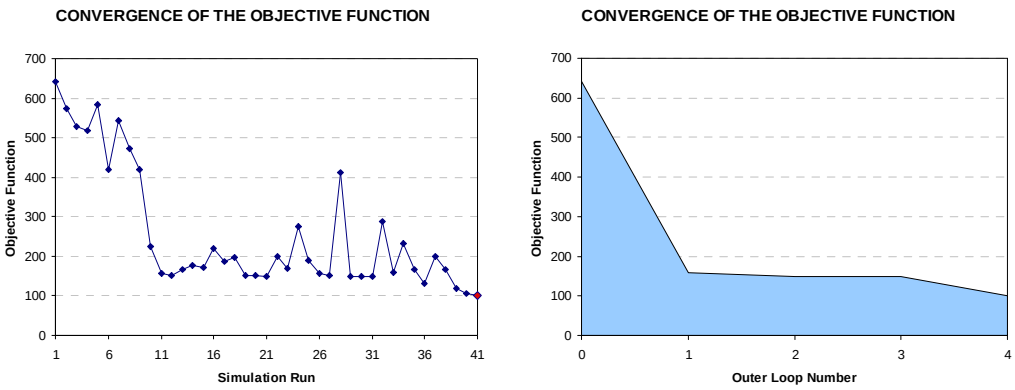
results were expected considering the limited number of conditioning data and simplifying assumptions employed to generate the variogram-based heterogeneity model and the petrophysical characterization.

8.4 RESULTS AND DISCUSSION

The distribution of connectivity indexes in layer 3 and 7 of the initial model are shown in Figure 8-8. Characteristic features of the spatial distribution of connectivity indexes include high anisotropy with a main direction of 15-25° azimuth, variations in relative proportions of connectivity indexes from sand to sand, with clear predominance of connectivity indexes 3 and 4 in upper sands and an increased proportion of connectivity indexes 1 and 2 in lower sands. The simulated production response of the initial model is shown in Figure 8-9. Significant deviations from the production history are evident in a faster depletion of the reservoir, higher field water cuts and clear mismatches of well water cuts. For example, at the end of the simulation the final pressure of the simulation response is roughly 800 psi below the production records, while the field water cut is almost double.

A systematic convergence of the objective function is observed through out the gradual deformation process. In the first outer iteration step the objective function drops from 642 to 148. In subsequent steps the convergence is more modest. The final

objective function value after 41 simulation runs is 100. The convergence is reasonable considering uncertainty in the completion and production history of at least two producers and an objective function composed of 8 different production variables and 172 time steps. The simulated production response matches fairly close the production history. The final model is reasonably good considering the simplifications and the uncertainty associated with the rock and fluid characterizations and the geological model. The initial and final realizations in the field-scale history matching study are shown in Figure 8-8. In comparison to the patchy distribution of rock types in the initial model, the final history matched model exhibits an improved continuity of rock types.



a) b)

Figure 8-7. Convergence characteristics of the history matching algorithm in the field case study: a) the fluctuation in objective function corresponding to all iteration steps during the process; b) The objective function of the updated models during the history matching process.

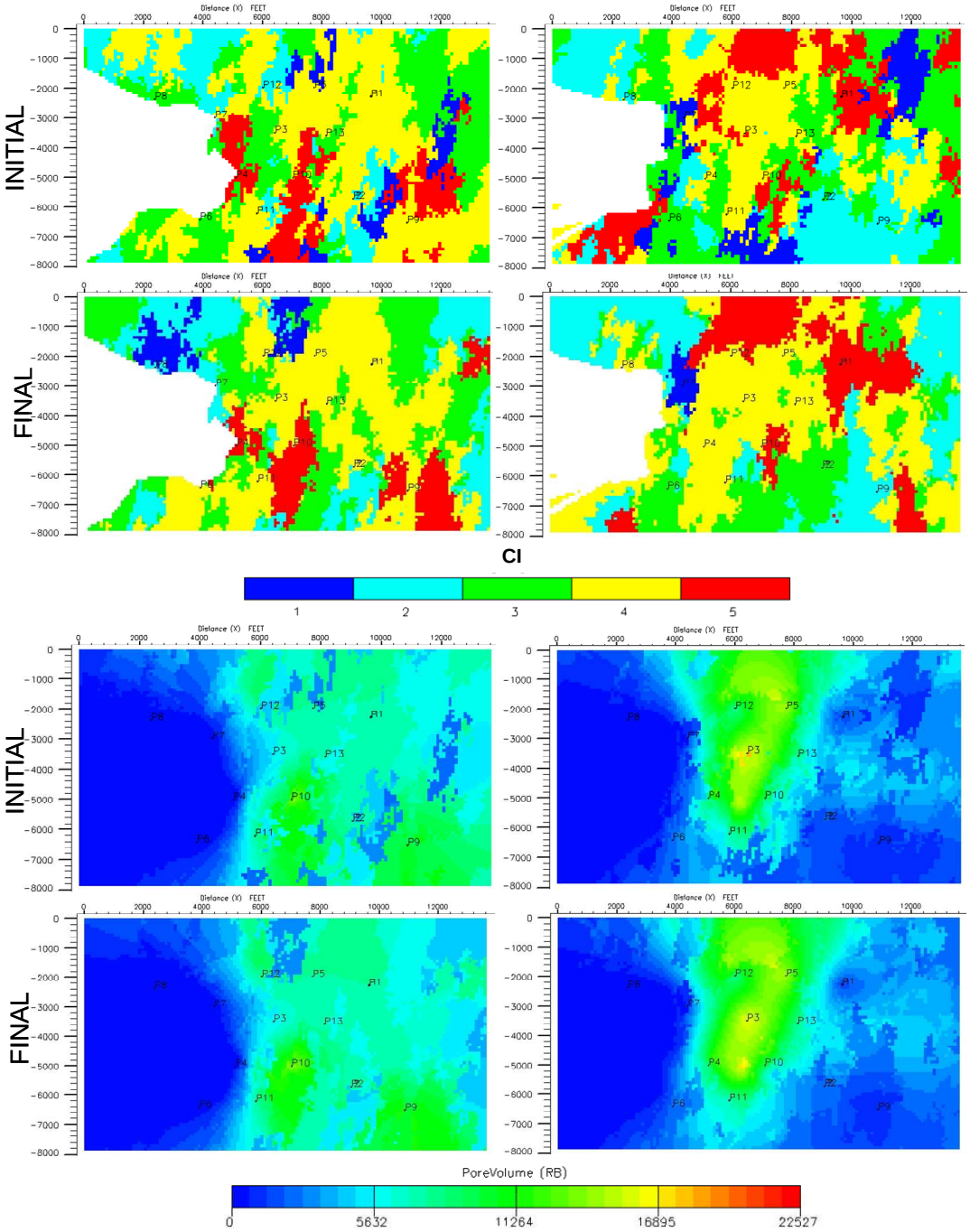


Figure 8-8. Connectivity index (top) and pore volume (bottom) distributions in layers 3 (left) and 7 (right) of the initial and final models corresponding to the field-scale history matching case study.

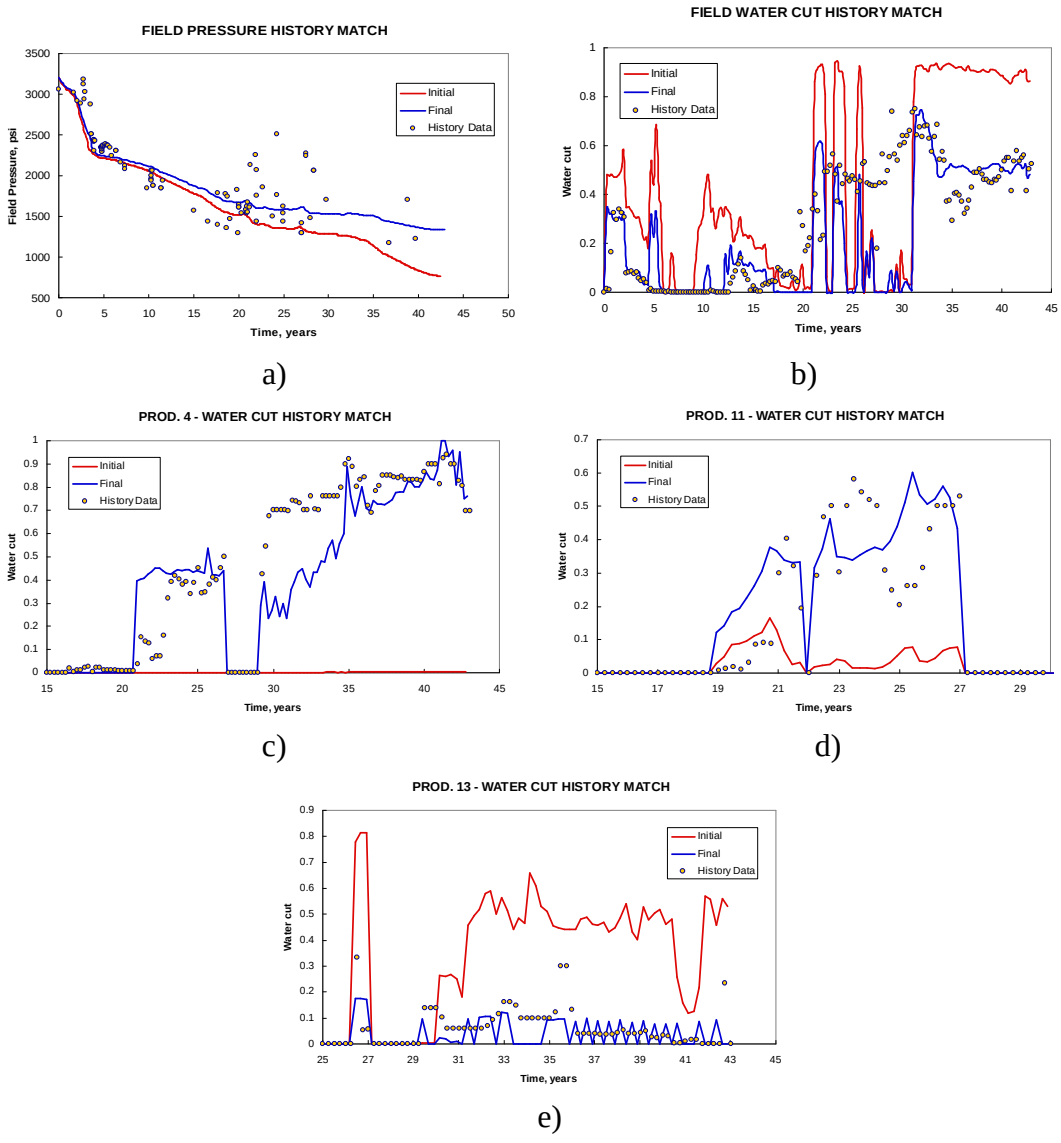


Figure 8-9. History match results: a) Field pressure match; b) Field water cut match; and well water cut match for producers c) 4; d) 11 and d) 13.

The results indicate that the proposed approach for the integration of dynamic data in the reservoir model and the consistent perturbation of all static and flow parameters is feasible for field-scale applications. Production variables included in

the objective function (water cuts and average pressure) are particularly sensitive to inter-well heterogeneity. Consequently, in this application the production data was particularly useful for calibrating the inter-well regions of the reservoir model.

A second history matching case was run to evaluate the importance of using sets of petrophysical properties consistent with the geological model. In contrast with the first case presented above, the second case considers a single set of multiphase flow functions for the entire reservoir. The flow functions corresponding to the rock type 4 were used in the second history matching case. The comparison of the result obtained after history matching using a single set of flow functions against that obtained using spatially varying flow functions, is shown in Figure 8-10 and Figure 8-11. The final model using geologically consistent multiphase flow functions (first case) showed better results in the convergence of the objective function and the match of the production history.

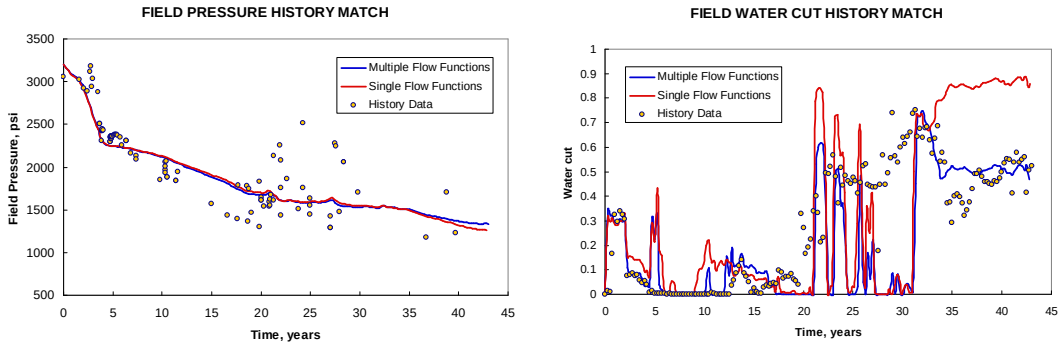


Figure 8-10 History match results for models with single and multiple sets of flow functions: Field pressure match (left) and field water cut match (right).

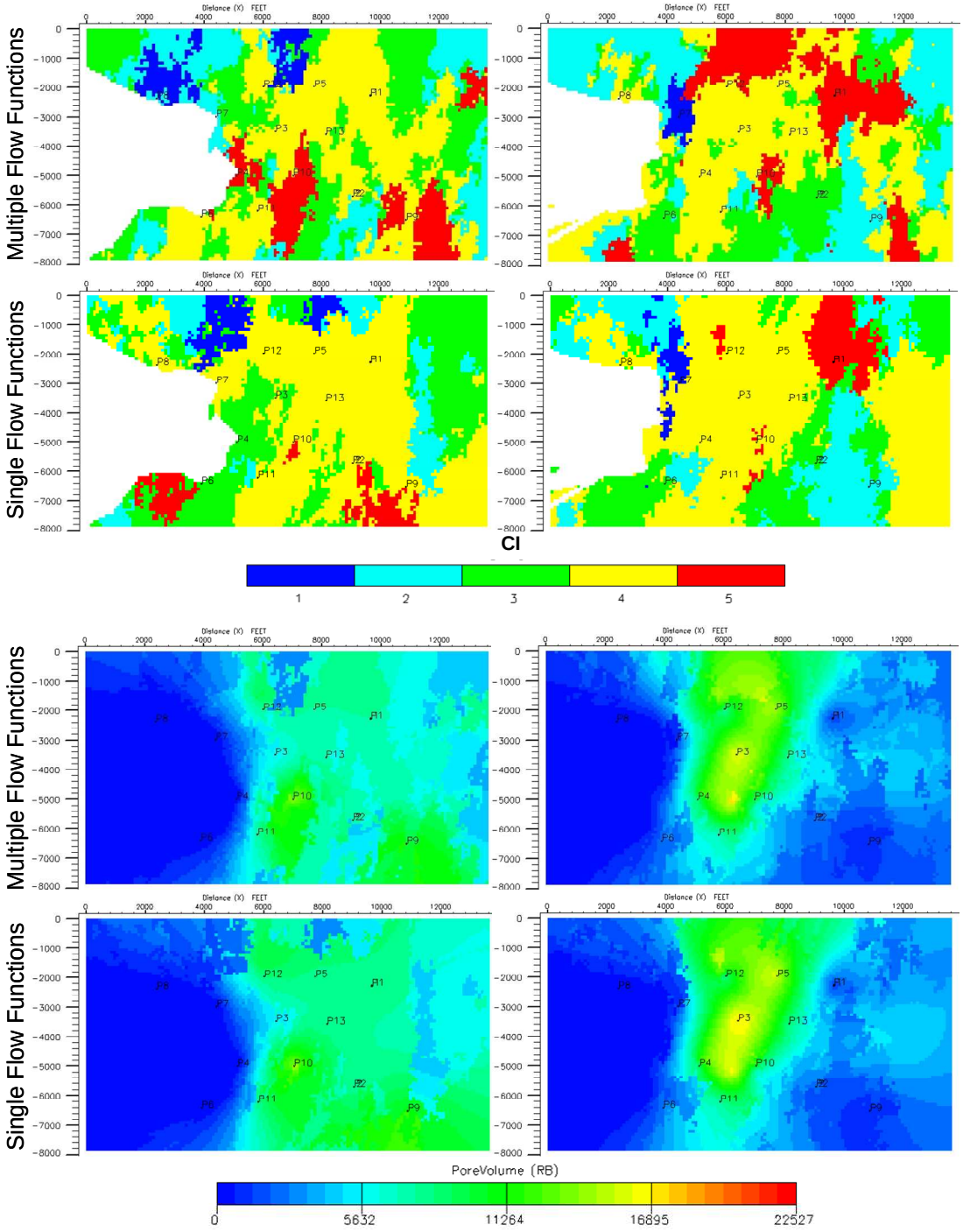


Figure 8-11 Connectivity index (top) and pore volume (bottom) distributions in layers 3 (left) and 7 (right) of the final history-matched models corresponding to the cases with single and multiple sets of flow function.

9 CONCLUSIONS AND RECOMMENDATIONS

The fundamental hypothesis tested in this dissertation is: “More reliable predictions of flow response could be expected from simulation models that consider the relationship between geological heterogeneity and multiphase flow functions.” The results and conclusions of this study support my assertion to believe that this hypothesis tested true.

9.1 KEY CONCLUSIONS

Several properties of rocks influence the displacement of fluids through the reservoir. Accurate prediction of performance of the reservoir requires accurate depiction of the spatial variability of all these flow properties. In this research, it is suggested that the spatial correlation of pores and throats at the pore-level influence the flow properties. Perturbation of these pore level spatial correlation characteristics consequently perturbs all the flow functions simultaneously. Since these spatial correlation characteristics at the pore level at a particular location in the reservoir are largely unknown, a scheme is presented that systematically perturbs the probability distributions characterizing that uncertainty.

The impact of pore structure characteristics on single and multiphase flow response of reservoir rocks is well known. However, the influence of heterogeneity at different scales and the multiple sources of uncertainty influencing the field-scale simulation results are frequently overlooked. Additionally, the lack of well-established techniques to acquire and analyze information describing structural properties of the pore space, render the relevance of this information in integrated reservoir characterization studies doubtful. A multi-scale approach was proposed to circumvent these setbacks and define a functional application of pore level descriptions to generate reservoir models that exhibit a more consistent relation between all the petrophysical properties, including static and multiphase flow characteristics. Practical approach to scale up the pore-level petrophysical functions to simulation grid blocks was attempted. This renders the multi-scale approach more practical.

Reliable pore level characterization techniques such as microtomography imaging are far from being considered standard procedures in reservoir characterization studies. In this dissertation we used pore network models combined with an analysis of microtomography reports available in the literature to overcome this common limitation. At the pore level, network models for different rock types are built considering characteristic properties of the basic geometric elements of pore structure and their correlation/cross-correlations. A program for the generation of

pore network models consistent with the pore structure described in microtomography studies has been developed. This algorithm considers characteristic distributions and correlations for the basic geometric elements of the pore structure (pore body size, throat size, throat length and connectivity) and their relation with basic parameter of the sedimentary rock (grain size distribution and sorting, porosity/compaction and cementation). Pore network models were qualitatively validated against some common characteristics of sedimentary rocks and general descriptions from tomography reports.

To obtain petrophysical properties representative of particular rock types, network models with appropriate representation of pore structure and a fluid invasion algorithm modeling the displacement processes at pore scale, are required. A pore-network simulator was developed to model primary drainage and imbibition displacement mechanisms under the assumptions of capillary control and no-gravity effects. The outputs of this simulator include the relative permeability and capillary pressure curves for both imbibition and drainage processes, the absolute permeability and porosity of the particular rock type.

Petrophysical properties including multiphase flow functions calculated from pore network models are representative for at most one or few inches. However, properties used in flow simulation models should represent a much larger scale, normally from dozens to hundreds of feet. To upscale the pore network properties to

grid block properties, a steady state approach is implemented considering mixtures of rock types described by appropriate spatial correlations and transition models. Each mixture of rock types is assigned a connectivity index and the procedure yields upscaled petrophysical properties corresponding to particular connectivity indexes. This approach also allows incorporating in the upscaled petrophysical properties, gravity effects that are ignored at the pore level. The reliability of the approach was tested with upscaling models considering different configurations for the distribution and transition of rock types.

Subtle spatial variations in permeability in a reservoir influence the flow of fluids in the reservoir thereby influencing the flow response recorded at the wells. This led us to postulate a probabilistic approach for dynamic data integration that hinges on the calibration of information contained in dynamic data and subsequently integrating that information with the prior geological knowledge about the reservoir. Different modifications of the gradual deformation approach for incorporating production data into reservoir models were implemented. These modifications basically render the gradual perturbation approach more efficient by increasing the range of variability during the optimization procedure and reducing the probability of getting trapped in local minima. The resultant, optimized probabilistic approach offers the great advantage of preserving the prior geological heterogeneity model during the integration of production data, resulting in more consistent models with improved accuracy for predicting the future production response of the reservoir.

At field scale, a stochastic simulation (sequential indicator simulation technique) is used to generate geologically consistent realizations of connectivity index distributions. A gradual deformation algorithm is used perturb the distribution of connectivity indexes to integrate the production data. This history matching algorithm combines the results of the pore network simulator and the upscaling technique to generate reservoir models with multiphase flow functions that are consistent with geological heterogeneity. For the history matching process, a Markov chain procedure was used to ensure global convergence during the history matching process. The proposed gradual deformation method renders the history match process faster and more controlled, increasing the consistency between the initial and the proposed realizations at every step, and improving the rate of convergence of the objective function.

Compared to other current techniques for history matching, the proposed method has the distinction of enabling consistent updates of all the flow functions while at the same time honoring the geological/sedimentary model for the distribution of petrophysical properties. Consequently, the reservoir model and its flow predictions are consistent with realistic geological settings. More importantly, it is conjectured that future predictions made using these updated models will be more reliable.

The proposed method for assisted history matching has been evaluated on different 3D cases with different geological complexity to identify the capability and limitations of the proposed approach. In the synthetic 3D cases, reservoir models with variable and complex heterogeneity were used as reference for generating the production history data. In these cases, the models generated during the history matching process differ from those used to generate the reference. The results provide interesting insights into the heterogeneity related information contained in dynamic data. The resulting history matched models show that the proposed approach is an efficient method to resolve the most relevant uncertainty in the inter-well regions of the reservoir by integrating production history data, resulting in more accurate future forecasts.

Variations in the flow simulation response of the reference reservoir models in the synthetic case study show the importance of using multiphase flow functions consistent with geological models. Results from the synthetic case study show the relevance of using multiphase flow functions consistent with geological models. The models for spatial variability of rock properties look quite different and more importantly, the model obtained by simultaneously perturbing all the flow functions yield more accurate prediction of the future performance of the reservoir.

The multi-scale petrophysically-consistent history matching approach was also implemented in a field case study. In a field case, the reliability of the prediction of future reservoir performance based on history-matched models is difficult to prove, since such data might not be available. However, the proposed approach offers two advantages: improved accuracy since the reservoir models are geologically consistent, and; an easy mechanism for updating the reservoir model as more production information become available. Results from this study proved that the proposed method is feasible for field-scale applications and lead to consistent reservoir models that produce more accurate predictions of flow response.

9.2 KEY LESSONS

- History matching is a complicated task that requires a thorough understanding of the production driving mechanisms, fluid behavior, rock-fluid interactions, geological structure and heterogeneity. Only an integrated assessment of all these characteristics will lead to reliable predictions.
- Important petrophysical properties including multiphase flow characteristics such as residual saturations, absolute and relative permeabilities are influenced by configuration of the pore structure.

- Pore network models represent a practical tool for evaluating the influence of the structural relationship and spatial correlation of geometric elements of pore space on single and multiphase flow characteristics.
- Heterogeneity at different scales plays an important role in the flow characteristics in reservoir rocks and consequently, should be considered in determining the representative flow functions for reservoir models.
- Multiphase flow functions can be upscaled from pore level to grid-block size or any other scale using a practical flow-based approach that is consistent with the geological heterogeneity at the final scale.
- The flow-based approach to upscale flow functions could be extended to a sequential upscaling procedure to consider characteristic heterogeneity models corresponding to progressively increasing scales.
- Considering the non-uniqueness of the history matching problem, validation of reservoir models should look beyond the reproduction of production history data, and focus on the accuracy in prediction of the future response.
- More accurate predictions of flow response can be obtained by ensuring the reproduction of the prior geological model of heterogeneity at all stages during the construction of reservoir models, including the integration of the production data.
- The probability perturbation approach implemented in this dissertation is a global-search method since is non-gradient based and considers a probabilistic

sampling scheme that avoids local minimum during the search of the optimum solution.

- The results of the case studies proved that the probability perturbation approach is a practical method for dynamic data integration through the calibration of the spatial distribution of connectivity indexes that encompass consistent petrophysical properties.

9.3 FUTURE WORK

The number of microtomography or similar studies that address a probabilistic characterization of the components of pore structure in rock samples is still limited in the literature. It would be wise to review the characteristic distributions and correlations used in the pore network algorithm to model the geometric elements of pore structure as more information becomes available.

The invasion algorithm, developed to model multiphase flow in pore networks, assumes water-wet, no-gravity effects, capillary controlled oil-water displacements. A lot of work can be done to extend this invasion algorithm to oil-gas and mixed-wet systems that consider also viscous or mixed (capillary-viscous) forces. Even though gravity effects can be incorporated in the petrophysical properties through the steady state upscaling approach, they could also be considered at the pore

level by implementing the appropriate modifications in the displacement mechanisms. Some of these implementations have already been reported in literature.

Different sensitivity studies were pursued with the pore network simulator to determine the influence of individual properties of rock texture and pore structure in the computed petrophysical properties of the network models. However, there are still a handful of studies that can be carried out to further analyze the factors that affect individual petrophysical properties. For example, the impact of cementation, anisotropy and compaction on static and multiphase flow properties of the rock could be studied.

The flow-based upscaling of petrophysical properties (including multiphase flow functions) can be used to evaluate the influence of heterogeneity at different scales. A sequential upscaling procedure to evaluate the influence of geological features at progressively larger scales could be implemented.

Pore network models can also be used to model fractured systems and carbonate rocks. However, these models would require major modifications of the pore network algorithm, if not a completely new development.

The probability perturbation approach was modified to increase the range of variation during the gradual deformation step to consider the transition between an

initial model and four different proposals instead of one. However there are still alternative schemes that could be applied to improve global search and convergence characteristics of this history matching method. For instance, strategies for pre-selection of proposals, gradual deformation between the initial model and optimum transition between proposals and improvements in the r_D spanning process, could be implemented.

The application of the multi-scale history matching approach was evaluated with synthetic and field case studies. However, a more robust validation scheme considering reservoirs with different geological environments, structural configurations, recovery mechanisms, fluid-rock properties and, development and production strategies; should be implemented to determine the general advantages, applicability and limitations of the multi-scale approach for history matching.

Appendix A: Pore Network Code

```
//Alvaro Barrera
//3D SQURE LATTICE NETWORK MODEL
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<math.h>
#include <stdio.h>
#include <time.h>

//*************** VARIABLES AND PARAMETERS ************************

const int MAXN = 110;  // Maximum Size N of the lattice NxNxN
const int MAXNN = MAXN*MAXN;  // NxN
const int MAXNNN = MAXN*MAXN*MAXN;  // NxNxN
const int MAXLN = 100;     // Maximum number of lines in the table for CDF input option
const double clsd = 1e-15;

int n, nn;   // Actual dimensions of the lattice
int band, nl;    // Actual Band size of conductance matrix and number of lines in CDF table
int inputMethod, inputParam;     // Input Method (1=Parameter File;2=CDF Table
File;3=Constant;4=Uniform Distrib.)
int n2; // Number of parameters 3*n*n*n - n*n
int blockedThr;    // Number of cement-blocked throats
int prt; // Print conductance/radii samples to file (yes = 1; no = 0)
double totalin, totalout;   // Tota rates in and out
double effectConduct, porosity; // Effectiv Conductivity and Porosity
double Vin, Vout;      // Inlet and Outlet potentials
double minParam, maxParam;    // parameters for constant and unifor distribution input options
double constant, exponent;   // Proportionality constant and exponent between conductance and
ratio/length
double cementThick; // Cement Thickness
char InputFile[40];     // Variable to store Parameter or CDF File name
double Conduc[MAXNNN+1][19]; // Mod for 18 bonds// Conductances assigned to each node in order
(i,j,k-1);(i,j-1,k);(i-1,j,k);(i+1,j,k);(i,j+1,k);(i,j,k+1)
double C[9*MAXNNN-13*MAXNN+6*MAXN+1];// Modified to consider 18 bonds// Initial
parameters (loaded or sampled) before process and assignment to nodes
double BC[MAXNNN+1], POT[MAXNNN+1]; // Boundary Conditions (RHS of conductance matrix)
double MB[MAXNNN+1];  // Mass Balance for each node
double RATE[MAXNNN+1][19]; //Modif for 18 bonds   // Rates calculated for each bond of each
node in order (i,j,k-1);(i,j-1,k);(i-1,j,k);(i+1,j,k);(i,j+1,k);(i,j,k+1)
double INLET[MAXNN+1];     // Inlet Rates
double OUTLET[MAXNN+1]; // Outlet Rates
double tr[MAXLN+1], pb[MAXLN+1];     // Throat radius and cumulative probabilities from original
input CDF
double TRTB[MAXLN+1], PTB[MAXLN+1];     // Throat radius and cumulative probabilities by
polynomial interpolation from CDF
```

```
int conn[MAXNNN+1];
double PoreBody[MAXNNN+1];
double Throat[MAXNNN+1][19];//Modif for 18 bonds
double Length[MAXNNN+1][19];//Modif for 18 bonds
double SWFN[101][4], SOF3[101][4];
int wct, occt;
bool imb;
double MaxCon;
double CorLengths[11];
char datafl[40];
int ixl, iyl, izl, ivrl, nx, ny, nz, ixv, rsrd;
double tmax, zmin, zmax, xmn, ymn, zmn, xsiz, ysiz, zsiz, aa1, aa2, radius, radius1, radius2;
int ndmax, nodmax, noct, istart, sstrat, mults, nmult;
double sang1, sang2, sang3;
const int MAXNST=4, MAXCUT= 11;
double c0[MAXCUT+1], cc[MAXNST*MAXCUT+1], aa[MAXNST*MAXCUT+1],
ang1[MAXNST*MAXCUT+1], ang2[MAXNST*MAXCUT+1], ang3[MAXNST*MAXCUT+1];
int it[MAXCUT*MAXNST+1], nst[MAXCUT+1];

void Conductance();  // Loads, samples or calculates the conductances for the bonds of each node
void ConductMatrix();    // Generates Conductance matrix on economic Storage
void BoundaryCond();    // Generates the Vector with boundary conditions (RHS with inlet/outle
potentials)
void LUDecomp();    // LU Decomposition of Conductance Matrix
void LUSolver(); // Solution of the node potentials.
void MassBalance(); // Mass Balance on each node and reports max and cum errors
void CDFTable();// Generates denser CDF table by 3-point polynomial interpolation
double PolyInterp(double ep, double x1, double x2, double x3, double y1, double y2, double y3);      //
Polynomial interpolation
double LinInterp(double ep, double x1, double x2, double y1, double y2);   // Linear Interpolation

void OldConnectivity();
const double PI = 3.14159265358979;
const double w = 1.9731;
const double tol = 0.01;
double InTens = 367.1; //mN/m-1
const double MinThroat = 10; //micro m
const double MaxThroat = 30000;   //micro m
const double MinConn = 2.0;
const double Viso = 0.8*0.001; //N-s/m2
const double Visw = 0.31*0.001; //N-s/m2
const double PcIncr = 10.0;  //N/m2 Capillary Pressure Increment
const double SwInitial = 0.21; // Initial Water Saturation for imbibition process
const int NCIs = 1;    // Number of different Connectivity Indexes
const int MaxComposed = 5;
double PBmeanLog[MaxComposed+1]; // Mean value of the Log of Pore Volumes Log(V)
double PBstdLog[MaxComposed+1]; // STD value of the distribution of Log of Pore Volumes Log(V)
double ThroatExp[MaxComposed+1];
double LengthExp[MaxComposed+1];
double CoordExp[MaxComposed+1];
double CoordThres[MaxComposed+1][11];
```

```
double NormArea[MaxComposed+1], Thress[MaxComposed+1], NormLength[MaxComposed+1],
C1[MaxComposed+1], C2[MaxComposed+1], TotalLength;
double AveGrainSize[MaxComposed+1] = {0, 20, 80.0, 0, 0, 0}; //Average Grain Size in micro m
double stdevGrainSize[MaxComposed+1] = {0, 20, 10.0, 0, 0, 0}; //Standard Deviation of Grain Size
in micro m
double AvePoro[MaxComposed+1] = {0, 0.25, 0.25, 0, 0, 0};     //Average Porosity of Rock
int CompBound[MaxComposed]= {0, 0, 0, 0, 0};  //Bounds between layers
double AvePoreVol[MaxComposed+1], StdevPoreVol[MaxComposed+1];
double DSat = 0.1;
int NComp = 2;    //Number of Layer on Composite model
int CompDir = 1; //Direction of layering 1= horizontal 2 = Vertical
int iComp;
int MixedComp = 1;  //Composite system with mixed properties = 1 (2 sources for now)
double MixCompFrac = 0.0;//Fraction of first rock type in mixed composite system
double CompFrac[MAXNNN+1];
double correlV, correlL;
double MaxError;
int Invaded[MAXNNN+1][20]; //Modif for 18 bonds
int List[MAXNNN+1], Path[MAXNNN+1];
double Pc, Volt, Volo, Sat, SoOld;
int tt;

void throats();
void sgsimParmFile(int nnx, int nny, int nnz);
void Connectivity();
int readparm();
void SuccOverRelaxation();
void QuickMassBalance();
int SearchIndex(int ind, int s);
void PrimaryDrainage();
void Imbibition();
bool CheckOilPath(int idd);
int CheckIndex(int idd);
void UpdateOilPath();
void WaterConduc();
bool checkBreakTrhough();
void OilConduc();
int copyfile();
void sgsimParmFilePB(double pbseed);
double getThroat(double PSize);
double getLength(double TSize);
double minn(double valA, double valB);
double maxx(double val1, double val2);
double clm;
double gauinv(double p);
double generate(double mean, double stdv);
void getiComp(int icp, int jcp, int kcp);

//********************** MAIN PROGRAM  ****************************

int main()
```

```
{
    int ici;
    ofstream ClearSWFN;
    ClearSWFN.open("SWFN.inc", ios::trunc);
    ClearSWFN<<endl<<"--Water saturation functions"<<endl;
    ClearSWFN<<"-- SWAT  KRW  PCOW"<<endl<<"SWFN";
    ClearSWFN.close();
    ofstream ClearSOF3;
    ClearSOF3.open("SOF3.inc", ios::trunc);
    ClearSOF3<<endl<<"--Oil saturation functions"<<endl;
    ClearSOF3<<"-- SOIL  KROW  KROG"<<endl<<"SOF3";
    ClearSOF3.close();
    CorLengths[1] = 4;
    CorLengths[2] = 9;
    CorLengths[3] = 10;
    CorLengths[4] = 11;
    CorLengths[5] = 12;
    CorLengths[6] = 13;
    CorLengths[7] = 14;
    CorLengths[8] = 15;
    readparm();
    for (ici=1;ici<=NCIs;ici++)
    {
        clm = CorLengths[ici];
       clock_t start, finish, fintemp;//Variable for computer running time
       double  duration;
       start = clock();// computer run start time
         Conductance();// read/sample/calculate/process/assign conductances to nodes
         OldConnectivity();
         fintemp = clock();// Computer run finish time
       duration = (double)(fintemp - start) / CLOCKS_PER_SEC;
       cout<<"Running Time = " <<duration<<endl<<endl;//Export computer running time
         BoundaryCond();// Build RHS with Boundary Conditions
       finish = clock();// Computer run finish time
       duration = (double)(finish - fintemp) / CLOCKS_PER_SEC;
       cout<<"Running Time = " <<duration<<endl<<endl;//Export computer running time
         SuccOverRelaxation();// Solve by Successive Over - Relaxation Iterative method
       fintemp = clock();// Computer run finish time
       duration = (double)(fintemp - finish) / CLOCKS_PER_SEC;
       cout<<"Running Time = " <<duration<<endl<<endl;//Export computer running time
       finish = clock();// Computer run finish time
       duration = (double)(finish - fintemp) / CLOCKS_PER_SEC;
       cout<<"Running Time = " <<duration<<endl<<endl;//Export computer running time
         MassBalance();// Mass Ballance
         PrimaryDrainage();
         Imbibition();
       fintemp = clock();// Computer run finish time
       duration = (double)(fintemp - finish) / CLOCKS_PER_SEC;
       cout<<"Running Time = " <<duration<<endl<<endl;//Export computer running time
       finish = clock();// Computer run finish time
       duration = (double)(finish - start) / CLOCKS_PER_SEC;
```

```
        cout<<endl<<"Total Computer Running Time = " <<duration<<endl<<endl;//Export computer
running time
    }
    ofstream EndSWFN;
    EndSWFN.open("SWFN.inc", ios::ate);
    EndSWFN<<endl<<"ENDINC";
    EndSWFN.close();
    ofstream EndSOF3;
    EndSOF3.open("SOF3.inc", ios::ate);
    EndSOF3<<endl<<"ENDINC";
    EndSOF3.close();
    ofstream resout;
    resout.open("Results.txt", ios::trunc);
    resout<<"Model Length = "<<TotalLength<<endl;
    resout.close();
    return 0;
}

//***** INPUT - SAMPLE - CALCULATIONS - ASSIGNMENT OF CONDUCTANCES ******

void Conductance()
{
    int i, j, k, index, indx, indy, indz, ci, pt1, incr, npb;
    double r, ep, x1, x2, y1, y2, PBave, PBstdev, sumpb;
    char tempch[150];
    cout<<"Assingning Conductances: "<<endl;
    if (inputMethod==0) // Spatial correlation pore/pore pore/throat
    {
        PBstdev = 2.0;
        incr = 0;
        while (fabs(1.0-PBstdev)>0.001)
        {
            PBave = 0.0;
            PBstdev = 0.0;
            sgsimParmFilePB(rsrd+incr);
            system("sgsim sgsim1.par");
            ifstream PBin("Porebody.out", ios::nocreate);
            if (!PBin)
            {
                cout<<"The pore body sample file does not exist"<<endl;
                exit(1);
            }
            for (i=1;i<=3;i++)
                PBin.getline(tempch,150);
            for (i=1;i<=nn;i++)
            {
                PBin>>PoreBody[i];
                PBave += PoreBody[i];
            }
            PBave = PBave/double(nn);
            PBin.close();
```

```
            for (i=1;i<=nn;i++)
            {
                PBstdev += pow(PoreBody[i] - PBave,2.0);
            }
            PBstdev = sqrt(PBstdev/double(nn-1));
            incr += 7;
        }
        for (i=1;i<=nn;i++)
        {
            PoreBody[i] -= PBave;
        }
        cout<<"Gausian Distribution ("<<PBave<<","<<PBstdev<<")"<<endl;
        ofstream PBout("GaussianDist.txt");
        PBout<<"Gaussian Distribution"<<endl<<1<<endl<<"Sample"<<endl;
        for (i=1;i<=nn;i++)
        {
            PBout<<PoreBody[i]<<endl;
        }
        PBout.close();
        for (iComp=1;iComp<=NComp;iComp++)
        {
            AveGrainSize[iComp] = AveGrainSize[iComp]/1000.0;
            stdevGrainSize[iComp] = stdevGrainSize[iComp]/1000.0;
            AvePoreVol[iComp] = 4.0*PI*pow(AveGrainSize[iComp]/5.0,3.0)/3.0;
            StdevPoreVol[iComp]
=4.0*PI/6.0*(pow((AveGrainSize[iComp]+stdevGrainSize[iComp])/5.0,3.0)-
pow((AveGrainSize[iComp]-stdevGrainSize[iComp])/5.0,3.0));
            PBstdLog[iComp] =
sqrt(log10(pow(StdevPoreVol[iComp],2.0)/pow(AvePoreVol[iComp],2.0)+1.0));
            PBmeanLog[iComp] = log10(AvePoreVol[iComp])-pow(PBstdLog[iComp],2)/2.0;
        }
        for (i=1;i<=nn;i++)
        {
            r = rand();// random generator
            r = (r/RAND_MAX);
            CompFrac[i] = r;
        }
        for (i=1;i<=n;i++)
        {
            for (j=1;j<=n;j++)
            {
                for (k=1;k<=n;k++)
                {
                    index = (k-1)*n*n + (j-1)*n + i;
                    getiComp(i, j, k);
                    PoreBody[index] = pow(10.0, PBmeanLog[iComp] +
PBstdLog[iComp]*PoreBody[index]);
                    PoreBody[index] = 1000.0*pow(3.0*PoreBody[index]/(4.0*PI),1.0/3.0);
                }
            }
        }
```

```
if (MixedComp==1)
{
    for (i=1;i<=n;i++)
    {
        for (j=1;j<=n;j++)
        {
            for (k=1;k<=n;k++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                sumpb = 4*PoreBody[index];
                npb = 4;
                if (i>1)
                {
                    indx = (k-1)*n*n + (j-1)*n + i - 1;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                if (i<n)
                {
                    indx = (k-1)*n*n + (j-1)*n + i + 1;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                if (j>1)
                {
                    indx = (k-1)*n*n + (j-2)*n + i;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                if (j<n)
                {
                    indx = (k-1)*n*n + j*n + i;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                if (k>1)
                {
                    indx = (k-2)*n*n + (j-1)*n + i;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                if (k<n)
                {
                    indx = k*n*n + (j-1)*n + i;
                    sumpb += PoreBody[indx];
                    npb += 1;
                }
                MB[index] = sumpb/npb;
            }
        }
    }
```

```
    }
    for (i=1;i<=nn;i++)
        PoreBody[i] = MB[i];
}
if (inputMethod==1) // parameter loaded from file
{
    ifstream fin(InputFile, ios::nocreate);// open file and read parameters
    if (!fin)
    {
        cout<<"There is no Input File for Conductances/Radius/Porebodies"<<endl;
        exit(1);
    }
    for (i=1;i<=3;i++)
        fin.getline(tempch,150);
    if (inputParam!=3)
    {
        for (i=1;i<=n2;i++)
        {
            fin>>C[i];
            if (C[i]<=0) cout<<"Possible Wrong Conductance "<<i<<" : "<<C[i]<<endl;
        }
    }
    fin.close();
    if (inputParam==3)
    {
        ifstream checkin("Porebody.out", ios::nocreate);// open file and read parameters
        if (!checkin)
        {
            cout<<"Please Rename the file "<<InputFile<<" to Porebody.out"<<endl;
            exit(1);
        }
        checkin.close();
    }
}
if (inputMethod==2) // parameters to be sampled from cdf table
{
    ifstream fin(InputFile, ios::nocreate);    // open file and read cdf table
    if (!fin)
    {
        cout<<"There is no CDF file"<<endl;
        exit(1);
    }
    fin>>nl;// number of rows in CDF table
    for(i=1;i<=nl;i++)     // read original CDF table
    {
        fin>>tr[i];
        fin>>pb[i];
    }
   fin.close();
    ofstream fout6;// Export original table to output file
    fout6.open("CumulativeDistr.txt");
```

```
    fout6<<"Original Cumulative Distribution"<<endl;
    for(i=1;i<=nl;i++)
    {
        fout6<<tr[i]<<"  ";
        fout6<<pb[i]<<endl;
    }
    CDFTable();// Generate Denser cdf table by 3-point polynomial interpolation
    fout6<<endl<<endl<<"Cumulative Distribution Search Table"<<endl;
    for(i=1;i<=100;i++)  // Export CDF table from polynomial interpolation to output file
    {
        fout6<<TRTB[i]<<"  ";
        fout6<<PTB[i]<<endl;
    }
    fout6.close();
    for (i=1;i<=n2;i++)   // Sample parameters form denser CDF table
    {
        r = rand();// random generator
        r = (r/RAND_MAX);
        pt1 = 1;
        while (r>PTB[pt1])
            pt1++;
        pt1 --;
        if (pt1>99)
        {
            C[i] = TRTB[100];
        }
        else
        {
            if (pt1<1)
            {
                C[i] = TRTB[1];
            }
            else
            {
                ep = r;
                x1 = PTB[pt1];
                x2 = PTB[pt1+1];
                y1 = TRTB[pt1];
                y2 = TRTB[pt1+1];
                C[i] = LinInterp(ep, x1, x2, y1, y2);//Linear interpolation from dense cdf Table
            }
        }
    }
}
if (inputMethod==3) //Constant parameter value for all bonds
{
    for (i=1;i<=n2;i++)
    {
        C[i] = minParam;
    }
}
```

```
if (inputMethod==4) // Parameter sampled from uniform distribution
{
    for (i=1;i<=n2;i++)
    {
        r = rand();
        r = (r/RAND_MAX)*(maxParam-minParam) + minParam;
        C[i] = r;
    }
}
blockedThr = 0;// Initialize number of cement-blocked throats
if (inputParam==2)    // If input parameter type is radius
{
    if (cementThick>0.0)// Correct radius for cement thickness
    {
        for (i=1;i<=n2;i++)
        {
            C[i] = C[i] - cementThick;
            if (C[i]<=0.0) // check for blocked throats
            {
                C[i] = clsd;
                blockedThr++;
            }
        }
    }
    if (prt>0)
    {
        ofstream fout1;// Export all loaded/sampled throat radius
        fout1.open("RadiusSample.txt");
       fout1<<"All Radii/Width"<<endl<<"1"<<endl;
       fout1<<"Radius"<<endl;
        for (i=1;i<=n2;i++)
        {
            fout1<<C[i]<<endl;
        }
        fout1.close();
    }
}
if (inputParam==3)    // If input parameter type is Pore body size
{
    throats();
    if (cementThick>0.0)// Correct radius for cement thickness
    {
        for (i=1;i<=nn;i++)
        {
            for (j=1; j<=18; j++) //Modif. for 18 bonds
            {
                if (Throat[i][j]>0.0)
                {
                    Throat[i][j] = Throat[i][j] - cementThick;
                    if (Throat[i][j]<=0.0) // check for blocked throats
                    {
```

```
                        Throat[i][j] = clsd;
                        blockedThr++;
                    }
                }
            }
        }
    }
    for (i=1;i<=nn;i++)
    {
        for (j=1; j<=18; j++) //Modif. for 18 bonds
        {
            if (Throat[i][j]>clsd)
                Conduc[i][j] = constant*pow(Throat[i][j],exponent)/Length[i][j];
            else
                Conduc[i][j] = clsd;
        }
    }
}
if (inputParam!=3)    // If input parameter type is radius
{
    ci = 1;
    for (k=1; k<=n; k++)
    {
        for (j=1; j<=n; j++)   // Assign the conductances to the nodes
        {
            for (i=1; i<=n; i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                Conduc[index][9] = C[ci];
                ci++;
                if (i==1)
                {
                    Conduc[index][2] = 0.0;
                    Conduc[index][6] = 0.0;
                    Conduc[index][11] = 0.0;
                    Conduc[index][15] = 0.0;
                }
                else
                {
                    indx = (k-1)*n*n + (j-1)*n + i - 1;
                    Conduc[indx][10] = Conduc[index][9];
                    if (k!=1)
                    {
                        Conduc[index][2] = C[ci];
                        ci++;
                        indx = (k-2)*n*n + (j-1)*n + i - 1;
                        Conduc[indx][17] = Conduc[index][2];
                    }
                    else
                        Conduc[index][2] = 0.0;
                    if (j!=1)
```

```
    {
        Conduc[index][6] = C[ci];
        ci++;
        indx = (k-1)*n*n + (j-2)*n + i - 1;
        Conduc[indx][13] = Conduc[index][6];
    }
    else
        Conduc[index][6] = 0.0;
    if (j!=n)
    {
        Conduc[index][11] = C[ci];
        ci++;
        indx = (k-1)*n*n + j*n + i - 1;
        Conduc[indx][8] = Conduc[index][11];
    }
    else
        Conduc[index][11] = 0.0;
    if (k!=n)
    {
        Conduc[index][15] = C[ci];
        ci++;
        indx = k*n*n + (j-1)*n + i - 1;
        Conduc[indx][4] = Conduc[index][15];
    }
    else
        Conduc[index][15] = 0.0;
}
if (i==n)
{
    Conduc[index][10] = C[ci];
    ci++;
    Conduc[index][17] = 0.0;
    Conduc[index][13] = 0.0;
    Conduc[index][8] = 0.0;
    Conduc[index][4] = 0.0;
}
if (j==1)
{
    Conduc[index][7] = 0.0;
    Conduc[index][8] = 0.0;
    Conduc[index][1] = 0.0;
    Conduc[index][14] = 0.0;
}
else
{
    Conduc[index][7] = C[ci];
    ci++;
    indy = (k-1)*n*n + (j-2)*n + i;
    Conduc[indy][12] = Conduc[index][7];
    if (k!=1)
    {
```

```
                    Conduc[index][1] = C[ci];
                    ci++;
                    indy = (k-2)*n*n + (j-2)*n + i;
                    Conduc[indy][18] = Conduc[index][1];
                }
                else
                    Conduc[index][1] = 0.0;
                if (k!=n)
                {
                    Conduc[index][14] = C[ci];
                    ci++;
                    indy = k*n*n + (j-2)*n + i;
                    Conduc[indy][5] = Conduc[index][14];
                }
                else
                    Conduc[index][14] = 0.0;
            }
            if (j==n)
            {
                Conduc[index][12] = 0.0;
                Conduc[index][13] = 0.0;
                Conduc[index][18] = 0.0;
                Conduc[index][5] = 0.0;
            }
            if (k==1)
            {
                Conduc[index][3] = 0.0;
                Conduc[index][4] = 0.0;
                Conduc[index][5] = 0.0;
            }
            else
            {
                Conduc[index][3] = C[ci];
                ci++;
                indz = (k-2)*n*n + (j-1)*n + i;
                Conduc[indz][16] = Conduc[index][3];
            }
            if (k==n)
            {
                Conduc[index][16] = 0.0;
                Conduc[index][17] = 0.0;
                Conduc[index][18] = 0.0;
            }
        }
    }
}
if (inputParam==2)
{
    for (i=1;i<=nn;i++)
    {
        for (j=1; j<=18; j++)
```

```
                {
                    if (Conduc[i][j]>clsd)
                        Conduc[i][j] = constant*pow(Conduc[i][j],exponent)/Length[i][j];
                }
            }
        }
    }

    if ((prt>0)&&(inputParam!=3))
    {
        ofstream fout2;// Export all conductances to file
        fout2.open("ConductanceSample.txt");
        fout2<<"All Conductances"<<endl<<"1"<<endl;
        fout2<<"Conductances"<<endl;
        for (i=1;i<=n2;i++)
        {
            fout2<<C[i]<<endl;
        }
        fout2.close();
    }
    return;
}

//************ CONNECTIVITY ALTERATION ******************************

void OldConnectivity()
{
    int i, id, k, j, ibo, m, opn, max, omax, count;
    double r, sum, aveConn, readin, readinn;
    int ind[19];
    int CC[19];
    int BondOrder[19];
    BondOrder[1] = 1;
    BondOrder[2] = 2;
    BondOrder[3] = 11;
    BondOrder[4] = 13;
    BondOrder[5] = 17;
    BondOrder[6] = 14;
    BondOrder[7] = 5;
    BondOrder[8] = 4;
    BondOrder[9] = 8;
    BondOrder[10] = 6;
    BondOrder[11] = 15;
    BondOrder[12] = 18;
    BondOrder[13] = 3;
    BondOrder[14] = 7;
    BondOrder[15] = 9;
    BondOrder[16] = 16;
    BondOrder[17] = 12;
    BondOrder[18] = 10;
    if (inputParam==3)
```

```
{
    Connectivity();
}
else
{
    for (i=1;i<=nn;i++)
    {
        r = rand();
        r = floor((r/RAND_MAX)*5 + 1.5);
        conn[i] = int(r);
    }
}
sum = 0;
for (i=1;i<=nn;i++)
{
    sum += conn[i];
    opn = 0;
    for(j=1;j<=18;j++)
    {
        if (Conduc[i][j]>clsd)
            opn += 1;
    }
    C[i] = 1.0*(opn - conn[i]);
}
aveConn = sum/nn;
for (k=1;k<=n;k++)
{
    for (j=1;j<=n;j++)     //   Loop over all nodes
    {
        for (i=1;i<=n;i++)
        {
            id = (k-1)*n*n + (j-1)*n + i;
            if (C[id]>0)
            {
                for (m=1;m<=18;m++)
                {
                    ind[m] = 0;
                    CC[m] = -18;
                }
                if ((k>1)&&(j>1)) ind[1] = (k-2)*n*n + (j-2)*n + i;
                if ((k>1)&&(i>1)) ind[2] = (k-2)*n*n + (j-1)*n + i - 1;
                if (k>1) ind[3] = (k-2)*n*n + (j-1)*n + i;
                if ((k>1)&&(i<n)) ind[4] = (k-2)*n*n + (j-1)*n + i + 1;
                if ((k>1)&&(j<n)) ind[5] = (k-2)*n*n + j*n + i;
                if ((i>1)&&(j>1)) ind[6] = (k-1)*n*n + (j-2)*n + i - 1;
                if (j>1) ind[7] = (k-1)*n*n + (j-2)*n + i;
                if ((i<n)&&(j>1)) ind[8] = (k-1)*n*n + (j-2)*n + i + 1;
                if (i>1) ind[9] = (k-1)*n*n + (j-1)*n + i - 1;
                if (i<n) ind[10] = (k-1)*n*n + (j-1)*n + i + 1;
                if ((i>1)&&(j<n)) ind[11] = (k-1)*n*n + j*n + i - 1;
                if (j<n) ind[12] = (k-1)*n*n + j*n + i;
```

```
                    if ((i<n)&&(j<n)) ind[13] = (k-1)*n*n + j*n + i + 1;
                    if ((k<n)&&(j>1)) ind[14] = k*n*n + (j-2)*n + i;
                    if ((k<n)&&(i>1)) ind[15] = k*n*n + (j-1)*n + i - 1;
                    if (k<n) ind[16] = k*n*n + (j-1)*n + i;
                    if ((k<n)&&(i<n)) ind[17] = k*n*n + (j-1)*n + i + 1;
                    if ((k<n)&&(j<n)) ind[18] = k*n*n + j*n + i;
                    max = 1;
                    for (ibo=1;ibo<=18;ibo++)
                    {
                        m = BondOrder[ibo];
                        if ((ind[m]>0)&&(Conduc[id][m]>clsd))
                        {
                            CC[m] = int(C[ind[m]]);
                            if (CC[m]>CC[max])
                                max = m;
                        }
                    }
                    while ((C[id]>0)&&(CC[max]>0))
                    {
                        if ((conn[ind[max]]>1)||((conn[ind[max]]==1)&&(CC[max]>1)))
                        {
                            Conduc[id][max] = clsd;
                            C[id] -= 1;
                            omax = 19-max;
                            Conduc[ind[max]][omax] = clsd;
                            CC[max] = -18;
                            C[ind[max]] -= 1;
                        }
                        else
                        {
                            CC[max] = -18;
                        }
                        for (ibo=1;ibo<=18;ibo++)
                        {
                            m = BondOrder[ibo];
                            if (CC[m]>CC[max])
                                max = m;
                        }
                    }
                }
            }
        }
    }
}
ofstream foutc;// Export all conductances to file
foutc.open("Connectivites.txt");
foutc<<"Average Connectivity: Target = "<<aveConn;
cout<<endl<<"Average Target Connectivity = "<<aveConn<<endl;
readin = 0.0;
readinn = 0.0;
count = 0;
sum = 0.0;
```

```
    for (i=1;i<=nn;i++)
    {
        readinn += 2.0*PoreBody[i];
        for(j=1;j<=18;j++)
        {
            if (Conduc[i][j]>clsd)
            {
                count += 1;
                readin += Length[i][j];
                sum += 1.0;
            }
        }
    }
    readinn = readinn/nn;
    readin = readin/double(count);
    TotalLength = n*(readinn+readin);
    aveConn = sum/nn;
    foutc<<"; Actual = "<<aveConn<<endl<<3<<endl;
    cout<<"Actual Average Connectivity = "<<aveConn<<endl;
    cout<<endl<<"Total Length = "<<TotalLength<<endl;
    foutc<<"Target"<<endl<<"Actual"<<endl<<"Offset"<<endl;
    for (i=1;i<=nn;i++)
    {
        foutc<<conn[i]<<"  "<<(conn[i]+C[i])<<"  "<<C[i]<<endl;
    }
    foutc.close();
    double minl, maxl, mint, maxt;
    if (prt>1)
    {
        ofstream Cout;
        ofstream CAout;
        Cout.open("AllocatedThroats.txt");
        CAout.open("AllThroats.txt");
        Cout<<"Throat Sizes Allocated to Each Pore Body"<<endl<<"8"<<endl;
        CAout<<"All Throat
Sizes"<<endl<<"3"<<endl<<"ThroatSize"<<endl<<"ThroatLength"<<endl<<"PoreSize"<<endl;

    Cout<<"BodySize"<<endl<<"CoordNumber"<<endl<<"MinThroatSize"<<endl<<"AveThroatSize
"<<endl;

    Cout<<"MaxThroatSize"<<endl<<"MinThroatLeng"<<endl<<"AveThroatLeng"<<endl<<"MaxTh
roatLeng"<<endl;
        for (i=1;i<=nn;i++)
        {
            minl = 10000.0;
            mint = 10000.0;
            maxl = 0.0;
            maxt = 0.0;
            count = 0;
            readin = 0.0;
            readinn = 0.0;
```

```
            Cout<<PoreBody[i]<<"  ";
            for (j=1; j<=18; j++)
            {
                if (Conduc[i][j]>clsd)
                {
                    CAout<<Throat[i][j]<<"  "<<Length[i][j]<<"  "<<PoreBody[i]<<endl;
                    if(Throat[i][j]>maxt)
                        maxt = Throat[i][j];
                    if(Throat[i][j]<mint)
                        mint = Throat[i][j];
                    if(Length[i][j]>maxl)
                        maxl = Length[i][j];
                    if(Length[i][j]<minl)
                        minl = Length[i][j];
                    readinn += Length[i][j];
                    readin += Throat[i][j];
                    count++;
                }
            }
            Cout<<count<<"  "<<mint<<"  "<<readin/count<<"  "<<maxt<<"  ";
            Cout<<minl<<"  "<<readinn/count<<"  "<<maxl<<endl;
        }
        Cout.close();
        CAout.close();
    }
    return;
}

//****** RHS - BOUNDARY CONDITIONS - INLET/OUTLET POTENTIALS ************

void BoundaryCond()
{
    int j, k, id, ind1, ind2;
    for (id=1;id<=nn;id++)   // Initialize RHS
    {
        BC[id] = 0.0;
    }
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)                                  // Include Inlet and Outlet potentials on
inlet/outlet nodes
        {
            ind1 = (k-1)*n*n + (j-1)*n + 1;
            BC[ind1] = Conduc[ind1][9]*Vin;
            ind2 = (k-1)*n*n + (j-1)*n + n;
            BC[ind2] = Conduc[ind2][10]*Vout;
        }
    }
    return;
}
```

```
//********  SOLVE SYSTEM OF EQUATIONS *************************

void SuccOverRelaxation()
{
    int index, i, j ,k, m, ct;
    double sum, old, dp;
    cout<<"Successive Over Relaxation Process"<<endl;
    dp = (Vin-Vout)/double(n);
    for (i=1;i<=n;i++)
    {
        for (j=1;j<=n;j++)
        {
            for (k=1;k<=n;k++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                POT[index] = Vin - double(i)*dp;
            }
        }
    }
    MaxError = 10.0*tol;
    ct = 0;
    do
    {
        ct++;
        for (k=1;k<=n;k++)
        {
            for (j=1;j<=n;j++)
            {
                for (i=1;i<=n;i++)
                {
                    index = (k-1)*n*n + (j-1)*n + i;
                    old = POT[index];
                    POT[index] = BC[index];
                    if ((k>1)&&(j>1))
                        POT[index] += Conduc[index][1]*POT[index-n*n-n];
                    if ((k>1)&&(i>1))
                        POT[index] += Conduc[index][2]*POT[index-n*n-1];
                    if (k>1)
                        POT[index] += Conduc[index][3]*POT[index-n*n];
                    if ((k>1)&&(i<n))
                        POT[index] += Conduc[index][4]*POT[index-n*n+1];
                    if ((k>1)&&(j<n))
                        POT[index] += Conduc[index][5]*POT[index-n*n+n];
                    if ((j>1)&&(i>1))
                        POT[index] += Conduc[index][6]*POT[index-n-1];
                    if (j>1)
                        POT[index] += Conduc[index][7]*POT[index-n];
                    if ((j>1)&&(i<n))
                        POT[index] += Conduc[index][8]*POT[index-n+1];
                    if (i>1)
                        POT[index] += Conduc[index][9]*POT[index-1];
```

```
                    if (i<n)
                        POT[index] += Conduc[index][10]*POT[index+1];
                    if ((j<n)&&(i>1))
                        POT[index] += Conduc[index][11]*POT[index+n-1];
                    if (j<n)
                        POT[index] += Conduc[index][12]*POT[index+n];
                    if ((j<n)&&(i<n))
                        POT[index] += Conduc[index][13]*POT[index+n+1];
                    if ((k<n)&&(j>1))
                        POT[index] += Conduc[index][14]*POT[index+n*n-n];
                    if ((k<n)&&(i>1))
                        POT[index] += Conduc[index][15]*POT[index+n*n-1];
                    if (k<n)
                        POT[index] += Conduc[index][16]*POT[index+n*n];
                    if ((k<n)&&(i<n))
                        POT[index] += Conduc[index][17]*POT[index+n*n+1];
                    if ((k<n)&&(j<n))
                        POT[index] += Conduc[index][18]*POT[index+n*n+n];
                    sum = 0.0;
                    for (m=1; m<=18; m++)
                        sum += Conduc[index][m];
                    POT[index] = (1-w)*old + w*POT[index]/sum;
                }
            }
        }
        QuickMassBalance();
    }
    while ((MaxError>tol)&&(ct<10000));
    cout<<"Number of Iterations = "<<ct<<endl;
    cout<<"Maximum Error = "<<MaxError<<endl;
    for (i=1;i<=nn;i++)   // Initialize RHS
    {
        BC[i] = POT[i];
    }
//  The solution of the system (node potentials) is on B - Export to file
        ofstream fout3;
        fout3.open("SolutionPotentials.txt");
        fout3<<"Node Potentials V"<<endl<<"1"<<endl<<"Potentials"<<endl;

        for (i=1;i<=nn;i++)
        {
            fout3<<POT[i]/6894.757<<endl;
        }
        fout3.close();
    return;
}

//*************** QUICK MATERIAL BALANCE *********************************

void QuickMassBalance()
{
```

```
int i, j, k, index, Max;
double sum;
for (k=1;k<=n;k++)
{
    for (j=1;j<=n;j++)     // Calculate rate for each bond and mass balance for each node
    {
        for (i=1;i<=n;i++)
        {
            index = (k-1)*n*n + (j-1)*n + i;
            MB[index] = 0;
            if ((k>1)&&(j>1))
                MB[index] += Conduc[index][1]*(POT[index-n*n-n]-POT[index]);
            if ((k>1)&&(i>1))
                MB[index] += Conduc[index][2]*(POT[index-n*n-1]-POT[index]);

            if (k>1)
                MB[index] += Conduc[index][3]*(POT[index-n*n]-POT[index]);
            if ((k>1)&&(i<n))
                MB[index] += Conduc[index][4]*(POT[index-n*n+1]-POT[index]);

            if ((k>1)&&(j<n))
                MB[index] += Conduc[index][5]*(POT[index-n*n+n]-POT[index]);
            if ((j>1)&&(i>1))
                MB[index] += Conduc[index][6]*(POT[index-n-1]-POT[index]);
            if (j>1)
                MB[index] += Conduc[index][7]*(POT[index-n]-POT[index]);
            if ((j>1)&&(i<n))
                MB[index] += Conduc[index][8]*(POT[index-n+1]-POT[index]);
            if (i>1)
                MB[index] += Conduc[index][9]*(POT[index-1]-POT[index]);
            if (i<n)
                MB[index] += Conduc[index][10]*(POT[index+1]-POT[index]);
            if ((j<n)&&(i>1))
                MB[index] += Conduc[index][11]*(POT[index+n-1]-POT[index]);
            if (j<n)
                MB[index] += Conduc[index][12]*(POT[index+n]-POT[index]);
            if ((j<n)&&(i<n))
                MB[index] += Conduc[index][13]*(POT[index+n+1]-POT[index]);
            if ((k<n)&&(j>1))
                MB[index] += Conduc[index][14]*(POT[index+n*n-n]-POT[index]);
            if ((k<n)&&(i>1))
                MB[index] += Conduc[index][15]*(POT[index+n*n-1]-POT[index]);

            if (k<n)
                MB[index] += Conduc[index][16]*(POT[index+n*n]-POT[index]);
            if ((k<n)&&(i<n))
                MB[index] += Conduc[index][17]*(POT[index+n*n+1]-POT[index]);

            if ((k<n)&&(j<n))
                MB[index] += Conduc[index][18]*(POT[index+n*n+n]-POT[index]);
            if (i==1)
```

```
                    MB[index] += Conduc[index][9]*(Vin-POT[index]);
                if (i==n)
                    MB[index] += Conduc[index][10]*(Vout-POT[index]);
            }
        }
    }
    Max = 1;                                    // Determine maximum and cumulative mass balance
error
    sum = 0.0;
    for (i=1;i<=nn;i++)
    {
        sum += fabs(MB[i]);
        if (fabs(MB[i])>fabs(MB[Max])) Max = i;
    }
    MaxError = fabs(MB[Max]);
    return;
}

//** MASS BALANCE - BOND AND INLET/OUTLET RATES - OTHER NETWORK MODEL
RESULTS *******

void MassBalance()
{
    int i, j, k, index, Max;
    double sum;
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)     // Calculate rate for each bond and mass balance for each node
        {
            for (i=1;i<=n;i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                MB[index] = 0;
                if ((k>1)&&(j>1))
                {
                    RATE[index][1] = Conduc[index][1]*(BC[index-n*n-n]-BC[index]);
                    MB[index] += RATE[index][1];
                }
                if ((k>1)&&(i>1))
                {
                    RATE[index][2] = Conduc[index][2]*(BC[index-n*n-1]-BC[index]);
                    MB[index] += RATE[index][2];
                }
                if (k>1)
                {
                    RATE[index][3] = Conduc[index][3]*(BC[index-n*n]-BC[index]);
                    MB[index] += RATE[index][3];
                }
                if ((k>1)&&(i<n))
                {
                    RATE[index][4] = Conduc[index][4]*(BC[index-n*n+1]-BC[index]);
```

```
    MB[index] += RATE[index][4];
}
if ((k>1)&&(j<n))
{
    RATE[index][5] = Conduc[index][5]*(BC[index-n*n+n]-BC[index]);
    MB[index] += RATE[index][5];
}
if ((j>1)&&(i>1))
{
    RATE[index][6] = Conduc[index][6]*(BC[index-n-1]-BC[index]);
    MB[index] += RATE[index][6];
}
if (j>1)
{
    RATE[index][7] = Conduc[index][7]*(BC[index-n]-BC[index]);
    MB[index] += RATE[index][7];
}
if ((j>1)&&(i<n))
{
    RATE[index][8] = Conduc[index][8]*(BC[index-n+1]-BC[index]);
    MB[index] += RATE[index][8];
}
if (i>1)
{
    RATE[index][9] = Conduc[index][9]*(BC[index-1]-BC[index]);
    MB[index] += RATE[index][9];
}
if (i<n)
{
    RATE[index][10] = Conduc[index][10]*(BC[index+1]-BC[index]);
    MB[index] += RATE[index][10];
}
if ((j<n)&&(i>1))
{
    RATE[index][11] = Conduc[index][11]*(BC[index+n-1]-BC[index]);
    MB[index] += RATE[index][11];
}
if (j<n)
{
    RATE[index][12] = Conduc[index][12]*(BC[index+n]-BC[index]);
    MB[index] += RATE[index][12];
}
if ((j<n)&&(i<n))
{
    RATE[index][13] = Conduc[index][13]*(BC[index+n+1]-BC[index]);
    MB[index] += RATE[index][13];
}
if ((k<n)&&(j>1))
{
    RATE[index][14] = Conduc[index][14]*(BC[index+n*n-n]-BC[index]);
    MB[index] += RATE[index][14];
```

```
            }
            if ((k<n)&&(i>1))
            {
                RATE[index][15] = Conduc[index][15]*(BC[index+n*n-1]-BC[index]);
                MB[index] += RATE[index][15];
            }
            if (k<n)
            {
                RATE[index][16] = Conduc[index][16]*(BC[index+n*n]-BC[index]);
                MB[index] += RATE[index][16];
            }
            if ((k<n)&&(i<n))
            {
                RATE[index][17] = Conduc[index][17]*(BC[index+n*n+1]-BC[index]);
                MB[index] += RATE[index][17];
            }
            if ((k<n)&&(j<n))
            {
                RATE[index][18] = Conduc[index][18]*(BC[index+n*n+n]-BC[index]);
                MB[index] += RATE[index][18];
            }
            if (i==1)
            {
                RATE[index][9] = Conduc[index][9]*(Vin-BC[index]);
                MB[index] += RATE[index][9];
                INLET[(k-1)*n + j] = RATE[index][9];
            }
            if (i==n)
            {
                RATE[index][10] = Conduc[index][10]*(Vout-BC[index]);
                MB[index] += RATE[index][10];
                OUTLET[(k-1)*n + j] = RATE[index][10];
            }
        }
    }
}
Max = 1;// Determine maximum and cumulative mass balance error
sum = 0.0;
for (i=1;i<=nn;i++)
{
    sum += fabs(MB[i]);
    if (fabs(MB[i])>fabs(MB[Max])) Max = i;
}
cout<<"Mass Balance"<<endl;
cout<<"Max Error : "<<MB[Max]<<", on node "<<Max<<endl;
cout<<"Average Error : "<<sum/nn<<endl;
cout<<"Cumulative Error : "<<sum<<endl<<endl;
if (prt>0)
{
    ofstream fout4;
    fout4.open("MassBalance.txt");// Output file for mass balance
```

```
        fout4<<"Mass Balance"<<endl<<endl;
// Export mass balance results to file
        fout4<<"Max Error : "<<MB[Max]<<", on node "<<Max<<endl;
        fout4<<"Average Error : "<<sum/nn<<endl;
        fout4<<"Cumulative Error : "<<sum<<endl<<endl;
        fout4.close();
    }
    totalin = 0.0;
    totalout = 0.0;
    for (i=1;i<=n*n;i++)
    {
        totalin += INLET[i];// Total inlet and outlet rates
        totalout -= OUTLET[i];
    }
    effectConduct = totalin*Visw*1E9/(TotalLength*(Vin-Vout));
    if (prt>0)
    {
       ofstream fout5;// Export inlet, outlet and bond rates
        fout5.open("BondRates.txt");
        fout5<<endl<<endl<<"Total In/Out"<<endl;// Export Total inlet and outlet rates
        fout5<<"Total In : "<<totalin<<"; Total Out : "<<totalout<<endl;
// Export effective conductance, porosity, cement-blocked throat ratio.
        fout5<<endl<<endl<<"Effective Network Conductance : "<<effectConduct<<endl;
        fout5<<"Blocked Throats by Cement : "<<blockedThr<<endl;
        fout5<<"Ratio of Blocked Throats : "<<blockedThr*1.0/n2<<endl;
        fout5.close();
    }
    cout<<"Total In : "<<totalin<<"; Total Out : "<<totalout<<endl;
    cout<<endl<<"Effective Network Conductance : "<<effectConduct<<endl;
    return;
}

//******** CDF SEARCH TABLE BY 3-POINT POLYNOMIAL INTERPOLATION *********

void CDFTable()
{
    int i, pt1, pt2, pt3, pt4, pt5, pt6;
    double rs, ep, x1, x2, x3, x4, x5, x6, y1, y2, y3, y4, y5, y6, ey1, ey2;
//
//  Generate Search Tables for Throat radius cumulative probability
//
    rs = (tr[nl]-tr[1])/99.0;
    for (i=1;i<=100;i++)
        TRTB[i] = tr[1] + (i-1)*rs;
    PTB[1] = pb[1];
    PTB[100] = pb[nl];
    for (i=2;i<=99;i++)
    {
        ep = TRTB[i];
        pt1 = 1;
        while (ep>tr[pt1])
```

```
            pt1++;
        pt1 --;
        pt2 = pt1 + 1;
        pt3 = pt1 + 2;
        pt4 = pt1 - 1;
        pt5 = pt1;
        pt6 = pt2;
        if (pt3>nl)
        {
            pt1--;
            pt2--;
            pt3--;
        }
        if (pt4<1)
        {
            pt4++;
            pt5++;
            pt6++;
        }
        x1 = tr[pt1];
        x2 = tr[pt2];
        x3 = tr[pt3];
        x4 = tr[pt4];
        x5 = tr[pt5];
        x6 = tr[pt6];
        y1 = pb[pt1]+1.;
        y2 = pb[pt2]+1.;
        y3 = pb[pt3]+1.;
        y4 = pb[pt4]+1.;
        y5 = pb[pt5]+1.;
        y6 = pb[pt6]+1.;
        ey1 = fabs(PolyInterp(ep, x1, x2, x3, y1, y2, y3) - 1.);
        ey2 = fabs(PolyInterp(ep, x4, x5, x6, y4, y5, y6) - 1.);
        PTB[i] = 0.5*ey1 + 0.5*ey2;
    }
    return;
}

//********** POLYNOMIAL AND LINEAR INTERPOLATORS *********************

double PolyInterp(double ep, double x1, double x2, double x3, double y1, double y2, double y3)
{
    double ye;
    ye = (ep-x2)*(ep-x3)*y1/((x1-x2)*(x1-x3)) + (ep-x1)*(ep-x3)*y2/((x2-x1)*(x2-x3)) + (ep-x1)*(ep-
x2)*y3/((x3-x1)*(x3-x2));
    return ye;
}

double LinInterp(double ep, double x1, double x2, double y1, double y2)
{
    double ye;
```

```
    ye = (ep-x2)*y1/(x1-x2) + (ep-x1)*y2/(x2-x1);
    return ye;
}

//** GENARATE THROAT SIZE DISTRIBUTION CORRELATED TO PORE SIZE
DISTRIBUTION ****

void throats()
{
    int i, j, k, index, indexx;
    double ThroatSize, BodyAve, ThroatLength, AGS, LPoro, Tres;
//
//  Initialize the Matrix with throat sizes
//
    for (iComp=1;iComp<=NComp;iComp++)
    {
        LengthExp[iComp] = -1.0;
        LPoro = AvePoro[iComp];
        AGS = 1000*AveGrainSize[iComp];
        while (LengthExp[iComp]<=0.0)
        {
            LengthExp[iComp] = 2*AGS*exp(-6.0*LPoro);
            LPoro -= 0.01;
        }
        ThroatExp[iComp] = 4100.0 + 9600*AvePoro[iComp];
        Tres = AGS/3.0;
        NormArea[iComp] = ThroatExp[iComp]/log(10.0)*(1.0-pow(10.0,-
32000.0/ThroatExp[iComp]));
        Thress[iComp] = pow(10.0,-Tres/LengthExp[iComp])/(2.0*Tres)*pow(Tres,2.0);
        NormLength[iComp] = Thress[iComp] + LengthExp[iComp]/log(10.0)*(pow(10.0,-
Tres/LengthExp[iComp])-pow(10.0,-1000.0/LengthExp[iComp]));
        Thress[iComp] = Thress[iComp]/NormLength[iComp];
        C1[iComp] = pow(10.0,-Tres/LengthExp[iComp]);
        C2[iComp] = C1[iComp]/Tres;
    }
    for (i=1;i<=nn;i++)
    {
        for (j=1; j<=18; j++)
            Throat[i][j] = 0.0;
            Length[i][j] = 0.0;
    }
//
//  ESTIMATION OF THROAT SIZES ON X DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on x Direction
//
    for (j=1; j<=n; j++)
    {
        for (k=1; k<=n; k++)
        {
            i = 1;
```

```
            getiComp(i, j, k);
            index = (k-1)*n*n + (j-1)*n + i;
            ThroatSize = getThroat(PoreBody[index]);
            Throat[index][9] = ThroatSize;
            ThroatLength = getLength(ThroatSize);
            Length[index][9] = ThroatLength;
            for (i=1; i<=(n-1); i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = (k-1)*n*n + (j-1)*n + i + 1;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j, k);
                else
                    getiComp(i+1, j, k);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][10] = ThroatSize;
                Throat[indexx][9] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][10] = ThroatLength;
                Length[indexx][9] = ThroatLength;
            }
            i = n;
            getiComp(i, j, k);
            index = (k-1)*n*n + (j-1)*n + i;
            ThroatSize = getThroat(PoreBody[index]);
            Throat[index][10] = ThroatSize;
            ThroatLength = getLength(ThroatSize);
            Length[index][10] = ThroatLength;
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON Y DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on y Direction
//
    for (i=1; i<=n; i++)
    {
        for (k=1; k<=n; k++)
        {
            for (j=1; j<=(n-1); j++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = (k-1)*n*n + j*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j, k);
                else
                    getiComp(i, j+1, k);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][12] = ThroatSize;
```

```
                Throat[indexx][7] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][12] = ThroatLength;
                Length[indexx][7] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON Z DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on z Direction
//
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
        {
            for (k=1; k<=(n-1); k++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = k*n*n + (j-1)*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j, k);
                else
                    getiComp(i, j, k+1);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][16] = ThroatSize;
                Throat[indexx][3] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][16] = ThroatLength;
                Length[indexx][3] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON YZ DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on yz Direction
//
    for (i=1; i<=n; i++)
    {
        for (k=1; k<=(n-1); k++)
        {
            for (j=1; j<=(n-1); j++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = k*n*n + j*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j, k);
                else
                    getiComp(i, j+1, k+1);
```

```
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][18] = ThroatSize;
                Throat[indexx][1] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][18] = ThroatLength;
                Length[indexx][1] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON +Y-Z DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on +y-z Direction
//
    for (i=1; i<=n; i++)
    {
        for (k=1; k<=(n-1); k++)
        {
            for (j=1; j<=(n-1); j++)
            {
                index = (k-1)*n*n + j*n + i;
                indexx = k*n*n + (j-1)*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j+1, k);
                else
                    getiComp(i, j, k+1);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][14] = ThroatSize;
                Throat[indexx][5] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][14] = ThroatLength;
                Length[indexx][5] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON XZ DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on xz Direction
//
    for (i=1; i<=(n-1); i++)
    {
        for (k=1; k<=(n-1); k++)
        {
            for (j=1; j<=n; j++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = k*n*n + (j-1)*n + i + 1;
                if (PoreBody[index]<=PoreBody[indexx])
```

```
                    getiComp(i, j, k);
                else
                    getiComp(i+1, j, k+1);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][17] = ThroatSize;
                Throat[indexx][2] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][17] = ThroatLength;
                Length[indexx][2] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON +X-Z DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on +x-z Direction
//
    for (i=1; i<=(n-1); i++)
    {
        for (k=1; k<=(n-1); k++)
        {
            for (j=1; j<=n; j++)
            {
                index = (k-1)*n*n + (j-1)*n + i + 1;
                indexx = k*n*n + (j-1)*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i+1, j, k);
                else
                    getiComp(i, j, k+1);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][15] = ThroatSize;
                Throat[indexx][4] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][15] = ThroatLength;
                Length[indexx][4] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON XY DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on xy Direction
//
    for (i=1; i<=(n-1); i++)
    {
        for (k=1; k<=n; k++)
        {
            for (j=1; j<=(n-1); j++)
            {
```

```
                index = (k-1)*n*n + (j-1)*n + i;
                indexx = (k-1)*n*n + j*n + i + 1;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i, j, k);
                else
                    getiComp(i+1, j+1, k);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][13] = ThroatSize;
                Throat[indexx][6] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][13] = ThroatLength;
                Length[indexx][6] = ThroatLength;
            }
        }
    }
//
//  ESTIMATION OF THROAT SIZES ON +X-Y DIRECTION:
//
//  Determine the average pore body size between adjacent pore bodies on +x-y Direction
//
    for (i=1; i<=(n-1); i++)
    {
        for (k=1; k<=n; k++)
        {
            for (j=1; j<=(n-1); j++)
            {
                index = (k-1)*n*n + (j-1)*n + i + 1;
                indexx = (k-1)*n*n + j*n + i;
                if (PoreBody[index]<=PoreBody[indexx])
                    getiComp(i+1, j, k);
                else
                    getiComp(i, j+1, k);
                BodyAve = minn(PoreBody[index],PoreBody[indexx]);
                ThroatSize = getThroat(BodyAve);
                Throat[index][11] = ThroatSize;
                Throat[indexx][8] = ThroatSize;
                ThroatLength = getLength(ThroatSize);
                Length[index][11] = ThroatLength;
                Length[indexx][8] = ThroatLength;
            }
        }
    }
    return;
}

//****************** GENERATE A PARAMETER FILE FOR SGSIM  **********************

void sgsimParmFilePB(double pbseed)
{
    ofstream Sout;
```

```
    Sout.open("sgsim1.par");
    Sout<<"                    Parameters for SGSIM"<<endl;
    Sout<<"                   **********************"<<endl<<endl;
    Sout<<"START OF PARAMETERS:"<<endl;
    Sout<<"cluster3D.dat"<<endl;
    Sout<<1<<"  "<<2<<"  "<<3<<"  "<<4<<"  "<<5<<"  "<<0<<endl;
    Sout<<-6.0<<"  "<<6.0<<endl;
    Sout<<0<<endl;
    Sout<<"sgsim.trn"<<endl;
    Sout<<0<<endl<<"RefPoreVolume.txt"<<endl;
    Sout<<1<<"  "<<2<<endl;
    Sout<<-6.0<<"  "<<6.0<<endl;
    Sout<<1<<"  "<<-6.0<<endl;
    Sout<<1<<"  "<<6.0<<endl;
    Sout<<0<<endl<<"sgsim.dbg"<<endl<<"Porebody.out"<<endl<<1<<endl;
    Sout<<n<<"  "<<0.5<<"  "<<1<<endl;
    Sout<<n<<"  "<<0.5<<"  "<<1<<endl;
    Sout<<n<<"  "<<0.5<<"  "<<1<<endl;
    Sout<<pbseed<<endl;
    Sout<<0<<"  "<<ndmax<<endl;
    Sout<<12<<endl;
    Sout<<1<<endl;
    Sout<<0<<"  "<<3<<endl;
    Sout<<0<<endl;
    Sout<<2.0*clm<<"  "<<2.0*clm<<"  "<<2.0*clm<<endl;
    Sout<<0<<"  "<<0<<"  "<<0<<endl;
    Sout<<41<<"  "<<41<<"  "<<31<<endl;
    Sout<<0<<"  "<<0.0<<"  "<<1.0<<endl;
    Sout<<"BodySize.txt"<<endl<<1<<endl;
    Sout<<1<<"  "<<0.1<<endl;
        Sout<<1<<"  "<<0.9<<"  "<<0<<"  "<<0<<"  "<<0<<endl;
        Sout<<clm<<"  "<<clm<<"  "<<clm<<endl;
    Sout.close();
    return;
}

//************ ESTIMATE VARIABLE CONNECTIVITY ************************

void Connectivity()
{
    int i, j, k, ii, index;
    char tempch[150];
    double sumprob, pbvalue;
    double CoordProb[11];
//
//  Generate Probabilities for the Each coordination number
//
    for (iComp=1;iComp<=NComp;iComp++)
    {
        CoordExp[iComp] = 1.2 + AvePoro[iComp]*7.2;
        sumprob = 0.0;
```

```
        for (i=3;i<=10;i++)
        {
            CoordProb[i] = pow(10.0,-double(i)/CoordExp[iComp]);
            sumprob += CoordProb[i];
            if (i==4)
            {
                CoordProb[2] = CoordProb[i];
                sumprob += CoordProb[i];
            }
        }
        for (i=2;i<=9;i++)
        {
            if (i==2)
                CoordProb[i] = CoordProb[i]/sumprob;
            else
                CoordProb[i] = CoordProb[i-1] + CoordProb[i]/sumprob;
            CoordThres[iComp][i] = gauinv(CoordProb[i]);
        }
    }
    ifstream PBin2("GaussianDist.txt", ios::nocreate);
    for (i=1;i<=3;i++)
        PBin2.getline(tempch,150);
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)
        {
            for (i=1;i<=n;i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                getiComp(i, j, k);
                PBin2>>pbvalue;
                ii = 2;
                while ((ii<10)&&(pbvalue>CoordThres[iComp][ii]))
                    ii += 1;
                conn[index] = ii;
            }
        }
    }
    PBin2.close();
    return;
}

//****************** READ INPUT PARAMETER FROM FILES ******************

int readparm()
{
//
//   Read Input Parameters:
//
    ifstream fin1("Input.par", ios::nocreate);// Open Input file with model specifications
    if (!fin1)   // Check if file exists
```

```
    {
        cout<<"There is no Input Parameter file"<<endl;
        exit(1);
    }
    fin1>>n;// Read model specification: size
    nn = n*n*n;
    n2 = 9*n*n*n - 13*n*n + 6*n;//Modif for 18 bonds
    inputParam = 3;// read Input parameter type
    fin1>>Vin;// Read inlet and outlet potentials
    fin1>>Vout;
    Vin = Vin*6894.757;//Convert from psi to N/m2
    Vout = Vout*6894.757;//Convert from psi to N/m2
    inputMethod = 0;// Read input method, files and other specifications
    fin1>>cementThick;
    fin1>>prt;
    fin1>>rsrd;
    fin1.close();
    constant = PI/(1E9*8.*Visw);
    exponent = 4.0;
    InTens = InTens*1000.0;
    srand(rsrd);// Seed number for the random generator
    ofstream HDout;
    HDout.open("cluster3D.dat");
    HDout<<"Clustered Data for Pore Level Reconstruction"<<endl<<"6"<<endl;
    HDout<<"Xlocation"<<endl<<"Ylocation"<<endl<<"Zlocation"<<endl;
    HDout<<"Primary"<<endl<<"Secondary"<<endl<<"Weight"<<endl;
    HDout.close();
    return 0;
}

//***************** SEARCH PORE BODY INDEX **********************

int SearchIndex(int ind, int s)
{
    int index, x, y, z;
    z = int((ind-1)/(n*n)) + 1;
    y = int((ind-(z-1)*n*n-1)/n) + 1;
    x = ind - (z-1)*n*n - (y-1)*n;
    if ((s==1)&&(z>1)&&(y>1))
        index = ind - n*n - n;
    else if ((s==2)&&(z>1)&&(x>1))
        index = ind - n*n - 1;
    else if ((s==3)&&(z>1))
        index = ind - n*n;
    else if ((s==4)&&(z>1)&&(x<n))
        index = ind - n*n + 1;
    else if ((s==5)&&(z>1)&&(y<n))
        index = ind - n*n + n;
    else if ((s==6)&&(y>1)&&(x>1))
        index = ind - n - 1;
    else if ((s==7)&&(y>1))
```

```
        index = ind - n;
    else if ((s==8)&&(y>1)&&(x<n))
        index = ind - n + 1;
    else if ((s==9)&&(x>1))
        index = ind - 1;
    else if ((s==10)&&(x<n))
        index = ind + 1;
    else if ((s==11)&&(y<n)&&(x>1))
        index = ind + n - 1;
    else if ((s==12)&&(y<n))
        index = ind + n;
    else if ((s==13)&&(y<n)&&(x<n))
        index = ind + n + 1;
    else if ((s==14)&&(z<n)&&(y>1))
        index = ind + n*n - n;
    else if ((s==15)&&(z<n)&&(x>1))
        index = ind + n*n - 1;
    else if ((s==16)&&(z<n))
        index = ind + n*n;
    else if ((s==17)&&(z<n)&&(x<n))
        index = ind + n*n + 1;
    else if ((s==18)&&(z<n)&&(y<n))
        index = ind + n*n + n;
    else
        index = 0;
    return index;
}

//************* PRIMARY DRAINAGE ************************

void PrimaryDrainage()
{
    int i, j, k, ind, ind2, count, nnew, init, it;
    double rcalc;
    bool finish, brktr, BKPD;
    imb = false;
    BKPD = true;
   ofstream fout5;// Export inlet, outlet and bond rates
    fout5.open("RelPerms.txt", ios::trunc);
    fout5<<"Primary Drainage Relative Permeability Curves"<<endl<<endl;
    fout5<<"Fluid  WaterSat  Pc  EffectiveK  RateIn  RateOut"<<endl;
    fout5.close();
    ofstream Dout;// Export original table to output file
    Dout.open("DrainageMap.txt", ios::trunc);
    Dout.close();
    ofstream Pc1out;// Export original table to output file
    Pc1out.open("DrainagePc.txt");
    Pc1out<<"Primary Drainage Capillary Pressure Curve"<<endl<<endl;
    Pc1out<<"Sw  Pc"<<endl;
    brktr = false;
    SoOld = -1.0;
```

```
Volt = 0.0;
tt = 0;
for (i=1;i<=nn;i++)
{
    for (j=1;j<=18;j++)
    {
        Invaded[i][j] = 0;
    }
}
for (i=1;i<=nn;i++)
{
    Invaded[i][0] = 1;
    Volt += 4.0*PI*pow(PoreBody[i],3)/3.0;
    for (j=1;j<=18;j++)
    {
        if ((Invaded[i][j]==0)&&(Throat[i][j]>clsd))
        {
            Volt += PI*pow(Throat[i][j],2)*Length[i][j];
            tt++;
            Invaded[i][j] = 1;
            ind = SearchIndex(i,j);
            if (ind!=0)
            {
                Invaded[ind][19-j] = 1;
            }
        }
    }
    Invaded[i][19] = 0;
}
Pc = 0.0;
Volo = 0.0;
Sat = 0.0;
count = 0;
it = 0;
rcalc = MinThroat+1.0;
Pc = 2.0*InTens/(MaxThroat);
while (BKPD)
{
    Pc += PcIncr;
    rcalc = 2*InTens/Pc;
    finish = false;
    for (j=1;j<=n;j++)
    {
        for (k=1;k<=n;k++)
        {
            ind = (k-1)*n*n + (j-1)*n + 1;
            if (Throat[ind][9]>rcalc)
            {
                if (Invaded[ind][9]==1)
                {
                    Volo += PI*pow(Throat[ind][9],2)*Length[ind][9];
```

```
                it++;
                Invaded[ind][9]=2;
            }
            if (Invaded[ind][0]==1)
            {
                count++;
                List[count] = ind;
                Invaded[ind][0] = 2;
                Volo += 4.0*PI*pow(PoreBody[ind],3)/3.0;
            }
        }
    }
}
init = 1;
while (finish==false)
{
    nnew = 0;
    for (i=init;i<=count;i++)
    {
        ind = List[i];
        for (j=1;j<=18;j++)
        {
            if ((Invaded[ind][j]==1)&&(Throat[ind][j]>rcalc))
            {
                Invaded[ind][j] = 2;
                Volo += PI*pow(Throat[ind][j],2)*Length[ind][j];
                it++;
                ind2 = SearchIndex(ind,j);
                if (ind2!=0)
                {
                    Invaded[ind2][19-j] = 2;
                    if (Invaded[ind2][0]==1)
                    {
                        nnew += 1;
                        List[count+nnew] = ind2;
                        Invaded[ind2][0] = 2;
                        Volo += 4.0*PI*pow(PoreBody[ind2],3)/3.0;
                    }
                }
            }
        }
    }
    init = count + 1;
    count += nnew;
    if (nnew==0)
        finish = true;
}
Sat = Volo*(1.0-0.2*exp(-1E6*Pc/(2.0*InTens)))/Volt;
Pc1out<<1-Sat<<"  "<<Pc/6894.757<<endl;
if ((Sat-SoOld)>=(5*DSat))
{
```

```
            cout<<">>"<<endl;
            cout<<"Pc = "<<Pc/6894.757<<";  So = "<<Sat<<"  ";
            brktr = checkBreakTrhough();
            SoOld = Sat;
            WaterConduc();
            if (brktr)
                OilConduc();
            cout<<endl;
        }
        if ((1-Sat)<=SwInitial)
        {
            BKPD = false;
            cout<<"Pc = "<<Pc/6894.757<<";  So = "<<Sat<<"  ";
            brktr = checkBreakTrhough();
            SoOld = Sat;
            WaterConduc();
            if (brktr)
                OilConduc();
            cout<<endl;
        }
    }
    double VoloC;
    VoloC = Volo*(1.0-0.2*exp(-1E6*Pc/(2.0*InTens)));
    Pc1out<<endl<<"Invaded pores = "<<count<<"; Ratio = "<<double(count)/double(nn)<<endl;
    Pc1out<<"Invaded throats = "<<it<<"; Ratio = "<<double(it)/double(tt)<<endl;
    Pc1out<<"Total Volume = "<<Volt<<"; Oil Volume = "<<VoloC<<"; Water Saturation = "<<1-
VoloC/Volt<<endl;
    Pc1out<<"Min invaded throat size = "<<rcalc<<endl;
    Pc1out.close();
    ofstream Dout1;                         // Export original table to output file
    Dout1.open("DrainageMap.txt");
    Dout1<<"Invasion"<<endl<<1<<endl<<"Phase"<<endl;
    for (i=1;i<=nn;i++)
        Dout1<<Invaded[i][0]<<endl;
    Dout1.close();
    return;
}

//********************** IMBIBITION ******************************

void Imbibition()
{
    int i, ii, j, k, ind, ind2, ind3, count, nnew , init, tc, it1, it2;
    double rcalc, minpb;
    bool cont, brktr;//, control;
    imb = true;
    ofstream Iout;
    Iout.open("ImbibitionMap.txt", ios::trunc);
    Iout.close();
    ofstream Pc2out;                        // Export original table to output file
    Pc2out.open("ImbibitionPc.txt");
```

```
Pc2out<<"Imbibition Capillary Pressure Curve"<<endl<<endl;
Pc2out<<"rt  Sw  Pc"<<endl;
SWFN[1][1] = SwInitial-0.01;
SWFN[1][2] = 0.0;
SWFN[1][3] = Pc/6894.757+1.0;
SWFN[2][1] = SwInitial;
SWFN[2][2] = 0.0;
SWFN[2][3] = Pc/6894.757;
wct = 2;
occt = 0;
SoOld += 1.0;
count = 0;
it1 = 0;
it2 = 0;
UpdateOilPath();
minpb = 10000.0;
for (j=1;j<=n;j++)
{
    for (k=1;k<=n;k++)
    {
        ind = (k-1)*n*n + (j-1)*n + 1;
        if (PoreBody[ind]<minpb)
            minpb = PoreBody[ind];
        if ((Invaded[ind][0]==1)&&(Invaded[ind][19]==0))
        {
            count++;
            Invaded[ind][19]=1;
            List[count] = ind;
            for (i=1;i<=18;i++)
            {
                if (Invaded[ind][i]==2)
                {
                    it1++;
                    Volo -= PI*pow(Throat[ind][i],2)*Length[ind][i];
                    Invaded[ind][i] = 1;
                    ind2 = SearchIndex(ind,i);
                    if ((ind2!=0))
                    {
                        Invaded[ind2][19-i] = 1;
                    }
                }
            }
        }
    }
}
cout<<"Minimum Pore Body at Entry = "<<minpb<<endl;
while (Pc>0.0)
{
    rcalc = 2*InTens/Pc;
    cont = true;
    for (j=1;j<=n;j++)
```

```
{
    for (k=1;k<=n;k++)
    {
        ind = (k-1)*n*n + (j-1)*n + 1;
        if ((Invaded[ind][0]==2)&&(Invaded[ind][19]==0))
        {
            tc = 0;
            for (i=1;i<=18;i++)
            {
                if (Invaded[ind][i]==2)
                    tc++;
            }
            if (PoreBody[ind]<(rcalc/(1.0+0.25*(tc-1))))
            {
                if (conn[ind]==1)
                {
                    Volo -= 4.0*PI*pow(PoreBody[ind],3)/3.0;
                    count++;
                    Invaded[ind][0]=1;
                    Invaded[ind][19]=1;
                    List[count] = ind;
                    for (i=1;i<=18;i++)
                    {
                        if (Invaded[ind][i]==2)
                        {
                            it1++;
                            Invaded[ind][i] = 1;
                            Volo -= PI*pow(Throat[ind][i],2)*Length[ind][i];
                            ind2 = SearchIndex(ind,i);
                            if ((ind2!=0))
                            {
                                    Invaded[ind2][19-i] = 1;
                            }
                        }
                    }
                }
            }
            else
            {
                count++;
                Invaded[ind][19] = 1;
                List[count] = ind;
                for (i=1;i<=18;i++)
                {
                    if ((Invaded[ind][i]==2)&&(Throat[ind][i]<2.0*InTens/(Pc*3.0)))
                    {
                            if (conn[ind]==1)
                            {
                                it2++;
                                Volo -= PI*pow(Throat[ind][i],2)*Length[ind][i];
                                Invaded[ind][i] = 1;
```

```
                                ind2 = SearchIndex(ind,i);
                                if (ind2!=0)
                                {
                                    Invaded[ind2][19-i] = 1;
                                }
                            }
                        }
                    }
                }
            }
        }
}
init = 1;
while (cont)
{
    nnew = 0;
    for (i=init;i<=count;i++)
    {
        ind = List[i];
        if (Invaded[ind][0]==2)
        {
            tc = 0;
            for (ii=1;ii<=18;ii++)
            {
                if (Invaded[ind][ii]==2)
                    tc++;
            }
            if (PoreBody[ind]<(rcalc/(1.0+0.25*(tc-1))))
            {
                if (conn[ind]==1)
                {
                    Volo -= 4.0*PI*pow(PoreBody[ind],3)/3.0;
                    Invaded[ind][0]=1;
                    for (ii=1;ii<=18;ii++)
                    {
                        if (Invaded[ind][ii]==2)
                        {
                            it1++;
                            Volo -= PI*pow(Throat[ind][ii],2)*Length[ind][ii];
                            Invaded[ind][ii] = 1;
                            ind2 = SearchIndex(ind,ii);
                            if ((ind2!=0))
                            {
                                    Invaded[ind2][19-ii] = 1;
                            }
                        }
                    }
                }
            }
            else
            {
```

```
        for (ii=1;ii<=18;ii++)
        {
            if ((Invaded[ind][ii]==2)&&(Throat[ind][ii]<2.0*InTens/(Pc*3.0)))
            {
                    if (conn[ind]==1)
                    {
                        it2++;
                        Volo -= PI*pow(Throat[ind][ii],2)*Length[ind][ii];
                        Invaded[ind][ii] = 1;
                        ind2 = SearchIndex(ind,ii);
                        if (ind2!=0)
                        {
                            Invaded[ind2][19-ii] = 1;
                        }
                    }
            }
        }
    }
}
for (j=1;j<=18;j++)
{
    if (Invaded[ind][j]==1)
    {
        ind2 = SearchIndex(ind,j);
        if (ind2!=0)
        {
            if ((Invaded[ind2][0]==1)&&(Invaded[ind2][19]==0))
            {
                nnew++;
                Invaded[ind2][19]=1;
                List[count+nnew] = ind2;
                for (ii=1;ii<=18;ii++)
                {
                    if (Invaded[ind2][ii]==2)
                    {
                        it1++;
                        Volo -= PI*pow(Throat[ind2][ii],2)*Length[ind2][ii];
                        Invaded[ind2][ii] = 1;
                        ind3 = SearchIndex(ind2,ii);
                        if ((ind3!=0))
                        {
                            Invaded[ind3][19-ii] = 1;
                        }
                    }
                }
            }
            if ((Invaded[ind2][0]==2)&&(Invaded[ind2][19]==0))
            {
                tc = 0;
                for (ii=1;ii<=18;ii++)
                {
```

```
                    if (Invaded[ind2][ii]==2)
                        tc++;
                }
                if (PoreBody[ind2]<(rcalc/(1.0+0.25*(tc-1))))
                {
                    if (conn[ind2]==1)
                    {
                        Volo -= 4.0*PI*pow(PoreBody[ind2],3)/3.0;
                        nnew++;
                        Invaded[ind2][0]=1;
                        Invaded[ind2][19]=1;
                        List[count+nnew] = ind2;
                        for (ii=1;ii<=18;ii++)
                        {
                            if (Invaded[ind2][ii]==2)
                            {
                                it1++;
                                Volo -= PI*pow(Throat[ind2][ii],2)*Length[ind2][ii];
                                Invaded[ind2][ii] = 1;
                                ind3 = SearchIndex(ind2,ii);
                                if (ind3!=0)
                                {
                                        Invaded[ind3][19-ii] = 1;
                                }
                            }
                        }
                    }
                }
                else
                {
                    nnew++;
                    List[count+nnew] = ind2;
                    Invaded[ind2][19] = 1;
                    for (ii=1;ii<=18;ii++)
                    {
                        if
((Invaded[ind2][ii]==2)&&(Throat[ind2][ii]<2.0*InTens/(Pc*3.0)))
                        {
                                if (conn[ind2]==1)
                                {
                                    it2++;
                                    Volo -= PI*pow(Throat[ind2][ii],2)*Length[ind2][ii];
                                    Invaded[ind2][ii] = 1;
                                    ind3 = SearchIndex(ind2,ii);
                                    if (ind3!=0)
                                    {
                                        Invaded[ind3][19-ii] = 1;
                                    }
                                }
                        }
                    }
```

```
                                }
                            }
                        }
                    }
                }
            }
            init = count + 1;
            count += nnew;
            if (nnew==0)
                cont = false;
        }
        Sat = Volo*(1.0-0.2*exp(-1E6*Pc/(2.0*InTens)))/Volt;
        Pc2out<<rcalc<<"  "<<1-Sat<<"  "<<Pc/6894.757<<endl;
        if ((SoOld-Sat)>=DSat)
        {
            cout<<">>"<<endl;
            cout<<"Pc = "<<Pc/6894.757<<";  So = "<<Sat<<"  ";
            brktr = checkBreakTrhough();
            SoOld = Sat;
            WaterConduc();
            if (brktr)
                OilConduc();
            cout<<endl;
        }
        Pc -= PcIncr;
        if (Pc<0.0)
            Pc = 0.0;
        UpdateOilPath();
    }
    cout<<"PoreSize = "<<rcalc<<";  Pc = "<<(Pc+PcIncr)/6894.757<<";  So = "<<Sat<<"  ";
    brktr = checkBreakTrhough();
    SoOld = Sat;
    WaterConduc();
    if (brktr)
        OilConduc();
    cout<<endl;
    cout<<"Effec. Porosity = "<<Volt/pow(TotalLength,3)<<endl;
    double Volol;
    Volol = Volo*(1.0-0.2*exp(-1E6*Pc/(2.0*InTens)));
    Pc2out<<endl<<"Invaded pores = "<<count<<"; Ratio = "<<double(count)/double(nn)<<endl;
    Pc2out<<"Total Invaded throats = "<<it1+it2<<"; Ratio = "<<double(it1+it2)/double(tt)<<endl;
    Pc2out<<"Invaded Throats by Piston Like Disp. = "<<it1<<"; Ratio =
"<<double(it1)/double(it1+it2)<<endl;
    Pc2out<<"Invaded Throats by Snap-off = "<<it2<<"; Ratio =
"<<double(it2)/double(it1+it2)<<endl;
    Pc2out<<"Total Volume = "<<Volt<<"; Oil Volume = "<<Volol<<"; Water Saturation = "<<1-
Volol/Volt<<endl;
    Pc2out<<"Max invaded pore body size = "<<rcalc<<endl;
    Pc2out.close();
    ofstream Iout1;// Export original table to output file
    Iout1.open("ImbibitionMap.txt");
```

```
    Iout1<<"Imbibition"<<endl<<1<<endl<<"Phase"<<endl;
    for (i=1;i<=nn;i++)
        Iout1<<Invaded[i][0]<<endl;
    Iout1.close();
    wct++;
    SWFN[wct][1] = 1.0;
    SWFN[wct][2] = 1.0;
    SWFN[wct][3] = 0.0;
    occt++;
    SOF3[occt][1] = 0.0;
    SOF3[occt][2] = 0.0;
    SOF3[occt][3] = 0.0;
    for (i=occt-2;i>=1;i--)
    {
        if (SOF3[i][2]<0.0001)
        {
            SOF3[i][2] = 0.0001;
            if (SOF3[i][2]<=SOF3[i+1][2])
            {
                SOF3[i][2] = SOF3[i+1][2] + 0.0001;
            }
            SOF3[i][3] = SOF3[i][2];
        }
    }
    ofstream SWFNout;
    ofstream SOF3out;
    SWFNout.open("SWFN.inc", ios::ate);
    SOF3out.open("SOF3.inc", ios::ate);
    for (i=1;i<=wct;i++)
    {
        SWFNout<<endl;
        for (j=1;j<=3;j++)
            SWFNout<<SWFN[i][j]<<"  ";
    }
    SWFNout<<"/"<<endl<<endl;
    for (i=occt;i>=1;i--)
    {
        SOF3out<<endl;
        for (j=1;j<=3;j++)
            SOF3out<<SOF3[i][j]<<"  ";
    }
    SOF3out<<"/"<<endl<<endl;
    SWFNout.close();
    SOF3out.close();
    return;
}

//***************** GET PORE BODY INDEX ******************

int CheckIndex(int idd)
{
```

```
    int x, y, z;
    z = int((idd-1)/(n*n)) + 1;
    y = int((idd-(z-1)*n*n-1)/n) + 1;
    x = idd - (z-1)*n*n - (y-1)*n;
    return x;
}

//******************** GET OIL CONNECTED PATH ************************

void UpdateOilPath()
{
    int i, j, k, index, index2, counter, first, add;
    bool contin;
    for (i=1;i<=nn;i++)
        conn[i] = 0;
    counter = 0;
    for (j=1;j<=n;j++)
    {
        for (k=1;k<=n;k++)
        {
            index = (k-1)*n*n + (j-1)*n + n;
            if (Invaded[index][0]==2)
            {
                counter++;
                Path[counter] = index;
                conn[index] = 1;
            }
        }
    }
    if (counter>0)
    {
        first = 1;
        contin = true;
        while (contin)
        {
            add = 0;
            for (i=first;i<=counter;i++)
            {
                index = Path[i];
                for (j=1;j<=18;j++)
                {
                    if (Invaded[index][j]==2)
                    {
                        index2 = SearchIndex(index,j);
                        if ((index2!=0)&&(Invaded[index2][0]==2)&&(conn[index2]==0))
                        {
                            add++;
                            Path[counter+add] = index2;
                            conn[index2] = 1;
                        }
                    }
```

```
                }
            }
            first = counter + 1;
            counter += add;
            if (add==0)
                contin = false;
        }
    }
    return;
}

//*************** NETWORK CONDUCTIVITY TO WATER ************

void WaterConduc()
{
    int index, i, j ,k, m, ct, ind, ind1, ind2;
    double sum, old, dp, totalin, totalout, con;
    constant = PI/(1E9*8.*Visw);
    exponent = 4.0;
    for (ind=1;ind<=nn;ind++)
    {
        for (j=1; j<=18; j++)
        {
            ind2 = SearchIndex(ind,j);
            if (Throat[ind][j]>clsd)
            {
                if (Invaded[ind][j]==1)
                {
                    if (ind2!=0)
                    {
                        if((Invaded[ind][0]==1)&&(Invaded[ind2][0]==1))
                        {
                            Conduc[ind][j] = constant*pow(Throat[ind][j],exponent)/Length[ind][j];
                        }
                        else
                        {
                            Conduc[ind][j] = constant*pow((Throat[ind][j]/1.5),exponent)/Length[ind][j];
                        }
                    }
                    else
                    {
                        if(Invaded[ind][0]==1)
                        {
                            Conduc[ind][j] = constant*pow(Throat[ind][j],exponent)/Length[ind][j];
                        }
                        else
                        {
                            Conduc[ind][j] = constant*pow((Throat[ind][j]/1.5),exponent)/Length[ind][j];
                        }
                    }
                }
```

```
            if (Invaded[ind][j]==2)
            {
                Conduc[ind][j] = constant*pow((Throat[ind][j]/10.),exponent)/Length[ind][j];
            }
        }
        else
        {
            Conduc[ind][j] = clsd;
        }
    }
}
dp = (Vin-Vout)/double(n);
for (i=1;i<=n;i++)
{
    for (j=1;j<=n;j++)
    {
        for (k=1;k<=n;k++)
        {
            index = (k-1)*n*n + (j-1)*n + i;
            POT[index] = Vin - double(i)*dp;
        }
    }
}
for (i=1;i<=nn;i++)   // Initialize RHS
{
    BC[i] = 0.0;
}
for (k=1;k<=n;k++)
{
    for (j=1;j<=n;j++)     // Include Inlet and Outlet potentials on inlet/outlet nodes
    {
        ind1 = (k-1)*n*n + (j-1)*n + 1;
        BC[ind1] = Conduc[ind1][9]*Vin;
        ind2 = (k-1)*n*n + (j-1)*n + n;
        BC[ind2] = Conduc[ind2][10]*Vout;
    }
}
MaxError = 10.0*tol;
ct = 0;
do
{
    ct++;
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)
        {
            for (i=1;i<=n;i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                old = POT[index];
                POT[index] = BC[index];
```

```
                    if ((k>1)&&(j>1))
                        POT[index] += Conduc[index][1]*POT[index-n*n-n];
                    if ((k>1)&&(i>1))
                        POT[index] += Conduc[index][2]*POT[index-n*n-1];
                    if (k>1)
                        POT[index] += Conduc[index][3]*POT[index-n*n];
                    if ((k>1)&&(i<n))
                        POT[index] += Conduc[index][4]*POT[index-n*n+1];
                    if ((k>1)&&(j<n))
                        POT[index] += Conduc[index][5]*POT[index-n*n+n];
                    if ((j>1)&&(i>1))
                        POT[index] += Conduc[index][6]*POT[index-n-1];
                    if (j>1)
                        POT[index] += Conduc[index][7]*POT[index-n];
                    if ((j>1)&&(i<n))
                        POT[index] += Conduc[index][8]*POT[index-n+1];
                    if (i>1)
                        POT[index] += Conduc[index][9]*POT[index-1];
                    if (i<n)
                        POT[index] += Conduc[index][10]*POT[index+1];
                    if ((j<n)&&(i>1))
                        POT[index] += Conduc[index][11]*POT[index+n-1];
                    if (j<n)
                        POT[index] += Conduc[index][12]*POT[index+n];
                    if ((j<n)&&(i<n))
                        POT[index] += Conduc[index][13]*POT[index+n+1];
                    if ((k<n)&&(j>1))
                        POT[index] += Conduc[index][14]*POT[index+n*n-n];
                    if ((k<n)&&(i>1))
                        POT[index] += Conduc[index][15]*POT[index+n*n-1];
                    if (k<n)
                        POT[index] += Conduc[index][16]*POT[index+n*n];
                    if ((k<n)&&(i<n))
                        POT[index] += Conduc[index][17]*POT[index+n*n+1];
                    if ((k<n)&&(j<n))
                        POT[index] += Conduc[index][18]*POT[index+n*n+n];
                    sum = 0.0;
                    for (m=1; m<=18; m++)
                        sum += Conduc[index][m];
                    POT[index] = (1-w)*old + w*POT[index]/sum;
                }
            }
        }
        QuickMassBalance();
    }
    while ((MaxError>tol)&&(ct<10000));
    for (i=1;i<=nn;i++)   // Initialize RHS
    {
        BC[i] = POT[i];
    }
    totalin = 0.0;
```

```
    totalout = 0.0;
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)     // Calculate rate for each bond and mass balance for each node
        {
            index = (k-1)*n*n + (j-1)*n + 1;
            totalin += Conduc[index][9]*(Vin-BC[index]);
            index = (k-1)*n*n + (j-1)*n + n;
            totalout -= Conduc[index][10]*(Vout-BC[index]);
        }
    }
    con = totalin*Visw*1E9/(TotalLength*(Vin-Vout));
   ofstream fout5;// Export inlet, outlet and bond rates
    fout5.open("RelPerms.txt", ios::ate);
    fout5<<"Water:  "<<(1-Sat)<<"  "<<Pc/6894.757<<"  "<<con<<"  "<<totalin<<"
"<<totalout<<endl;
    fout5.close();
    if (Sat==0.0)
        MaxCon = con;
    cout<<"WIt="<<ct<<"  krw="<<con/MaxCon<<"  ";
    if ((imb)&&((1-Sat)>SwInitial))
    {
            wct++;
            SWFN[wct][1] = 1-Sat;
            SWFN[wct][2] = con/MaxCon;
            SWFN[wct][3] = Pc/6894.757;
        if (Pc==0.0)
            OilConduc();
    }
    return;
}

//********************** CHECK WATER OIL BREAKTHROUGH ********************

bool checkBreakTrhough()
{
    int index, j, k;
    bool ret;
    ret = false;
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)     // Calculate rate for each bond and mass balance for each node
        {
            index = (k-1)*n*n + (j-1)*n + n;
            if (Invaded[index][10]==2)
                ret = true;
        }
    }
    return ret;
}
```

```
//******************** NETWORK CONDUCTIVITY TO OIL ********************

void OilConduc()
{
    int index, i, j ,k, m, ct, ind, ind1, ind2;
    double sum, old, dp, totalin, totalout, con;
    constant = PI/(1E9*8.*Viso);
    exponent = 4.0;
    for (ind=1;ind<=nn;ind++)
    {
        for (j=1; j<=18; j++)
        {
            ind2 = SearchIndex(ind,j);
            if (Throat[ind][j]>clsd)
            {
                if (Invaded[ind][j]==2)
                {
                    if (ind2!=0)
                    {
                        if((Invaded[ind][0]==2)&&(Invaded[ind2][0]==2))
                        {
                            Conduc[ind][j] = constant*pow(Throat[ind][j],exponent)/Length[ind][j];
                        }
                        else
                        {
                            Conduc[ind][j] = constant*pow((Throat[ind][j]/1.5),exponent)/Length[ind][j];
                        }
                    }
                    else
                    {
                        if(Invaded[ind][0]==2)
                        {
                            Conduc[ind][j] = constant*pow(Throat[ind][j],exponent)/Length[ind][j];
                        }
                        else
                        {
                            Conduc[ind][j] = constant*pow((Throat[ind][j]/1.5),exponent)/Length[ind][j];
                        }
                    }
                }
                if (Invaded[ind][j]==1)
                {
                    Conduc[ind][j] = clsd;
                }
            }
            else
            {
                Conduc[ind][j] = clsd;
            }
        }
    }
```

```
dp = (Vin-Vout)/double(n);
for (i=1;i<=n;i++)
{
    for (j=1;j<=n;j++)
    {
        for (k=1;k<=n;k++)
        {
            index = (k-1)*n*n + (j-1)*n + i;
            POT[index] = Vin - double(i)*dp;
        }
    }
}
for (i=1;i<=nn;i++)   // Initialize RHS
{
    BC[i] = 0.0;
}
for (k=1;k<=n;k++)
{
    for (j=1;j<=n;j++)     // Include Inlet and Outlet potentials on inlet/outlet nodes
    {
        ind1 = (k-1)*n*n + (j-1)*n + 1;
        BC[ind1] = Conduc[ind1][9]*Vin;
        ind2 = (k-1)*n*n + (j-1)*n + n;
        BC[ind2] = Conduc[ind2][10]*Vout;
    }
}
MaxError = 10.0*tol;
ct = 0;
do
{
    ct++;
    for (k=1;k<=n;k++)
    {
        for (j=1;j<=n;j++)
        {
            for (i=1;i<=n;i++)
            {
                index = (k-1)*n*n + (j-1)*n + i;
                old = POT[index];
                POT[index] = BC[index];
                if ((k>1)&&(j>1))
                    POT[index] += Conduc[index][1]*POT[index-n*n-n];
                if ((k>1)&&(i>1))
                    POT[index] += Conduc[index][2]*POT[index-n*n-1];
                if (k>1)
                    POT[index] += Conduc[index][3]*POT[index-n*n];
                if ((k>1)&&(i<n))
                    POT[index] += Conduc[index][4]*POT[index-n*n+1];
                if ((k>1)&&(j<n))
                    POT[index] += Conduc[index][5]*POT[index-n*n+n];
                if ((j>1)&&(i>1))
```

```
                    POT[index] += Conduc[index][6]*POT[index-n-1];
                if (j>1)
                    POT[index] += Conduc[index][7]*POT[index-n];
                if ((j>1)&&(i<n))
                    POT[index] += Conduc[index][8]*POT[index-n+1];
                if (i>1)
                    POT[index] += Conduc[index][9]*POT[index-1];
                if (i<n)
                    POT[index] += Conduc[index][10]*POT[index+1];
                if ((j<n)&&(i>1))
                    POT[index] += Conduc[index][11]*POT[index+n-1];
                if (j<n)
                    POT[index] += Conduc[index][12]*POT[index+n];
                if ((j<n)&&(i<n))
                    POT[index] += Conduc[index][13]*POT[index+n+1];
                if ((k<n)&&(j>1))
                    POT[index] += Conduc[index][14]*POT[index+n*n-n];
                if ((k<n)&&(i>1))
                    POT[index] += Conduc[index][15]*POT[index+n*n-1];
                if (k<n)
                    POT[index] += Conduc[index][16]*POT[index+n*n];
                if ((k<n)&&(i<n))
                    POT[index] += Conduc[index][17]*POT[index+n*n+1];
                if ((k<n)&&(j<n))
                    POT[index] += Conduc[index][18]*POT[index+n*n+n];
                sum = 0.0;
                for (m=1; m<=18; m++)
                    sum += Conduc[index][m];
                POT[index] = (1-w)*old + w*POT[index]/sum;
            }
        }
    }
    QuickMassBalance();
}
while ((MaxError>tol)&&(ct<10000));
for (i=1;i<=nn;i++)   // Initialize RHS
{
    BC[i] = POT[i];
}
totalin = 0.0;
totalout = 0.0;
for (k=1;k<=n;k++)
{
    for (j=1;j<=n;j++)     // Calculate rate for each bond and mass balance for each node
    {
        index = (k-1)*n*n + (j-1)*n + 1;
        totalin += Conduc[index][9]*(Vin-BC[index]);
        index = (k-1)*n*n + (j-1)*n + n;
        totalout -= Conduc[index][10]*(Vout-BC[index]);
    }
}
```

```
    con = totalin*Viso*1E9/(TotalLength*(Vin-Vout));
    cout<<"OIt="<<ct<<"  kro="<<fabs(con/MaxCon);
   ofstream fout5;// Export inlet, outlet and bond rates
    fout5.open("RelPerms.txt", ios::ate);
    fout5<<"Oil:  "<<(1-Sat)<<"  "<<Pc/6894.757<<"  "<<con<<"  "<<totalin<<"  "<<totalout<<endl;
    fout5.close();
    if (imb)
    {
        if (occt==0)
        {
            occt++;
            SOF3[occt][1] = 1-SwInitial;
            SOF3[occt][2] = fabs(con/MaxCon);
            SOF3[occt][3] = SOF3[occt][2];
        }
        else
        {
            if ((1-Sat)>SwInitial)
            {
                occt++;
                SOF3[occt][1] = Sat;
                SOF3[occt][2] = fabs(con/MaxCon);
                SOF3[occt][3] = SOF3[occt][2];
            }
        }
        if (Pc==0.0)
        {
            SOF3[occt][1] = Sat;
            SOF3[occt][2] = 0.0;
            SOF3[occt][3] = SOF3[occt][2];
        }
    }
    return;
}

//************ CALCULATE THROAT SIZE ****************************

double getThroat(double PSize)
{
    double MaxArea, NormValue, rr, Const, ThSize;
    MaxArea = PI*pow(PSize,2);
    NormValue = ThroatExp[iComp]/log(10)*(1.0-pow(10,-MaxArea/ThroatExp[iComp]));
    rr = rand();// random generator
    rr = (rr/RAND_MAX);
    Const = rr*log(10)*NormValue/ThroatExp[iComp];
    ThSize = -ThroatExp[iComp]*log10(1-Const);
    ThSize = sqrt(ThSize/PI);
    return ThSize;
}

//****************** CALCULATE THROAT LENGTH ***************
```

```
double getLength(double TSize)
{
    double ThArea, ThProb, lProb, rr, lSize, nProb, AGS2;
    ThArea = PI*pow(TSize,2.0);
    ThProb = ThroatExp[iComp]/log(10.0)*(1.0-pow(10.0,-
ThArea/ThroatExp[iComp]))/NormArea[iComp];
    nProb = 2.0;
    while ((nProb>=1.0)||(nProb<=0.0))
    {
        rr = 2.0;
        while (fabs(rr)>1.0)
            rr = generate(0.0,0.1);
        nProb = ThProb + rr;
    }
    lProb = 1.0 - nProb;
    if (lProb<=Thress[iComp])
        lSize = sqrt(2.0*lProb*NormLength[iComp]/C2[iComp]);
    else
    {
        AGS2 = 1000*AveGrainSize[iComp]/3.0;
        lSize = -LengthExp[iComp]*log10(C1[iComp]-(lProb*NormLength[iComp]-
C2[iComp]*pow(AGS2,2.0)/2.0)*log(10.0)/LengthExp[iComp]);
    }
    if (lSize<1.0)
        lSize += 1.0;
    return lSize;
}

//***************** MINIMUM & MAXIMUM VALUE ******************************

double minn(double valA, double valB)
{
    double minVal;
    if (valA<=valB)
        minVal = valA;
    else
        minVal = valB;
    return minVal;
}

double maxx(double val1, double val2)
{
    double maxVal;
    if (val1>=val2)
        maxVal = val1;
    else
        maxVal = val2;
    return maxVal;
}
```

```
//************** GAUSSIAN INVERSION *******************************

double gauinv(double p)
{
/* Computes the inverse of the standard normal cumulative distribution
 function with a numerical approximation
   p    = double precision cumulative probability value: dble(psingle)
   xp   = G^-1 (p) in single precision
   ierr = 1 - then error situation (p out of range), 0 - OK
   real*8 p0,p1,p2,p3,p4,q0,q1,q2,q3,q4,yz,pp,lim,p
   save   p0,p1,p2,p3,p4,q0,q1,q2,q3,q4,lim
*/
     double p0, p1, p2, p3, p4, q0, q1, q2, q3, q4, yz, pp, lim, xp;
    lim = 1.0e-10;
     p0 = -0.322232431088;
     p1 = -1.0;
     p2 = -0.342242088547;
     p3 = -0.0204231210245;
     p4 = -0.0000453642210148;
     q0 = 0.0993484626060;
     q1 = 0.588581570495;
     q2 = 0.531103462366;
     q3 = 0.103537752850;
     q4 = 0.0038560700634;
// Check for an error situation:
     if (p<lim)
     {
          xp = -1.0e10;
          return xp;
     }
     if (p>(1.0-lim))
     {
          xp =  1.0e10;
          return xp;
     }
// Get k for an error situation:
     pp = p;
     if (p>0.5)
          pp = 1.0 - pp;
     xp = 0.0;
     if (p==0.5)
          return xp;
// Approximate the function:
     yz = sqrt(log(1.0/(pp*pp)));
     xp = double(yz + ((((yz*p4+p3)*yz+p2)*yz+p1)*yz+p0)/((((yz*q4+q3)*yz+q2)*yz+q1)*yz+q0));
     if (p==pp)
          xp = -xp;
// Return with G^-1(p):
     return xp;
}
```

```
//*********** DRAW FROM GAUSSIAN DIST ************************

double generate(double mean, double stdv)
{
    double result, r, rv;
    r = rand();// random generator
    r = (r/RAND_MAX);
    rv = gauinv(r);
    result = rv*stdv + mean;
    return result;
}

//****************** GET PORE BODY CONNECTED *********************

void getiComp(int icp, int jcp, int kcp)
{
    int frai;
    if (MixedComp==1)
    {
        frai = (kcp-1)*n*n + (jcp-1)*n + icp;
        if (CompFrac[frai]<=MixCompFrac)
            iComp = 1;
        else
            iComp = 2;
        return;
    }
    iComp = 1;
    if (NComp>1)
    {
        if (((CompDir==1)&&(kcp>CompBound[1]))||((CompDir==2)&&(icp>CompBound[1])))
        {
            iComp = 2;
            if (NComp>2)
            {
                if
(((CompDir==1)&&(kcp>CompBound[2]))||((CompDir==2)&&(icp>CompBound[2])))
                {
                    iComp = 3;
                    if (NComp>3)
                    {
                        if
(((CompDir==1)&&(kcp>CompBound[3]))||((CompDir==2)&&(icp>CompBound[3])))
                        {
                            iComp = 4;
                            if (NComp>4)
                            {
                                if
(((CompDir==1)&&(kcp>CompBound[4]))||((CompDir==2)&&(icp>CompBound[4])))
                                    iComp = 5;
                            }
                        }
```

```
                    }
                }
            }
        }
    }
    return;
}
```

Appendix B: Input File for Pore Network Code

Example of input files for pore network model

```
50
310
300
0.0
2
331231
```

Parameter description for input file:

50	=	50x50x50 lattice
310	=	Inlet Potential
300	=	Outlet Potential
0.0	=	Cement Thicknes (for Radius Input)
2	=	Output options
331231	=	Seed for rand number generator

Note: The pore network code requires the executable for the sequential gaussian simulator from the GSLIB library (developed in Stanford University).

Appendix C: History Matching Code

```
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<math.h>
#include <stdio.h>
#include <time.h>
#include <process.h>

const int MAXX=140, MAXY=80, MAXZ=15, MAXCTX= 71, MAXCTY=71, MAXCTZ=21,
MAXSBX=21, MAXSBY=21, MAXSBZ=11, MAXDAT=100000, MAXTAB=1250, MAXCUT= 11,
MAXNOD=25, MAXSAM=16, MAXNST=4;
const int MXYZ=MAXX*MAXY*MAXZ, MAXKR1=2*MAXNOD+2*MAXSAM+1,
MAXKR2=MAXKR1*MAXKR1, MAXROT=MAXCUT*MAXNST+1,
MAXCXY=MAXCTX*MAXCTY, MAXXYZ=MAXCTX*MAXCTY*MAXCTZ,
MAXSB=MAXSBX*MAXSBY*MAXSBZ, MXSXY=4*MAXSBX*MAXSBY,
MXSX=2*MAXSBX, MXCUT=MAXCUT+1;
double EPSLON=1.0e-20,UNEST=-99.0;
const int KORDEI=12, MAXOP1=KORDEI+1, MAXINT=int(pow(2,30));
int ixnode[MAXXYZ+1], iynode[MAXXYZ+1], iznode[MAXXYZ+1], nisb[MAXSB+1], inoct[8+1],
icnode[MAXNOD+1], ixsbtosr[8*MAXSB+1], iysbtosr[8*MAXSB+1], izsbtosr[8*MAXSB+1],
it[MAXCUT*MAXNST+1], nst[MAXCUT+1], ltail, middle, utail, sstrat, mults, nmult,
nviol[MAXCUT+1], ivtype, imbsim;
double ltpar, mpar, utpar, x[MAXDAT+1], y[MAXDAT+1], z[MAXDAT+1],
vr[MAXDAT+1][MXCUT+1], close[MAXDAT+1], actloc[MAXDAT+1], tmpdat[MAXDAT+1];
double sim[MXYZ+1], order[MXYZ+1], tmp[MXYZ+1], gcut[MAXTAB+1], gcdf[MAXTAB+1],
thres[MAXCUT+1], d[MAXCUT+1], cdf[MAXCUT+1], ccdf[MAXCUT+1], ccdfo[MAXCUT+1];
double beez[MAXCUT+1], c0[MAXCUT+1], cmax[MAXCUT+1], cc[MAXNST*MAXCUT+1],
aa[MAXNST*MAXCUT+1], ang1[MAXNST*MAXCUT+1], ang2[MAXNST*MAXCUT+1];
double ang3[MAXNST*MAXCUT+1], anis1[MAXNST*MAXCUT+1],
anis2[MAXNST*MAXCUT+1], aviol[MAXCUT+1], xviol[MAXCUT+1],
covtab[MAXCTX+1][MAXCTY+1][MAXCTZ+1][MAXCUT+1], cnodex[MAXNOD+1];
double cnodey[MAXNOD+1], cnodez[MAXNOD+1], cnodev[MAXNOD+1], cnodet[MAXNOD+1],
vra[MAXKR1+1];
double rotmat[MAXROT+1][3+1][3+1], r[MAXKR1+1], rr[MAXKR1+1], s[MAXKR1+1],
a[MAXKR2+1];
bool atnode[MAXDAT+1], softdat[MAXKR1+1];

const int MAXHMCOL=100, MAXHMRAW=1000; //SIZE OF MATRIX WITH PRODUCTION
HISTORY (VECTORS (COL) AND TIME STEPS (RAW))
const int MV=10;
int ivrs[MAXCUT+1];
int ncut, ixl, iyl, izl, ivrl, ixs, iys, izs, itabvr, itabwt, idbg, nx, ny, nz, nxy, nxyz, nd, ng;
int ixv[MAXOP1+1];
int ndmax, nodmax, maxsec, noct, mik, ktype, istart;
char temp[200];
char datafl[40], tabfl[40], softfl[40], outfl[40], dbgfl[40], str[40], title[80], PHFile[40];
bool testfl;
```

```
double var[MV], p, tmin, tmax, zmin, zmax, xmn, ymn, zmn, xsiz, ysiz, zsiz, aa1, aa2, radius, radius1,
radius2;
double radsqd, sanis1, sanis2, sang1, sang2, sang3, cutmik;

double c[MAXDAT+1], e[MAXDAT+1][MXCUT+1], f[MAXDAT+1], g[MAXDAT+1],
h[MAXDAT+1], ierr, sec1[MAXDAT+1], sec2[MAXDAT+1], sec3[MAXDAT+1];
int isrot, nhd, nlooku, nctx, ncty, nctz, nclose, ncnode;
int numcol, numraw;       //Number of production history variables to match (colums) and number of
values on time (raws)
double rd;

double HD[MXYZ+1];
int clss[MXYZ+1];
int nxsup, nysup, nzsup, nsbtosr;
double xmnsup, ymnsup, zmnsup, xsizsup, ysizsup, zsizsup;

double ProdHist[MAXHMRAW+1][MAXHMCOL+1]; // Matrix with production history data for
history match.
double SimProd[MAXHMRAW+1][MAXHMCOL+1]; // Matrix with simulated production history
data for history match.
double ProdHistVar[MAXHMCOL+1];

int RockType[MXYZ+1];
const int MaxSavedEval = 9;
double rdspan[MaxSavedEval+1] = { 0, 0.0, 0.0625, 0.25, 0.4375, 0.5, 0.5625, 0.75, 0.9375, 1.0};
double currentRDT;
int nSavedEval;
int maxIterInnerLoop;
int maxIterOuterLoop;
const int Realizations = 1;

double objFunction[MaxSavedEval+1];
double colObjFunct[MaxSavedEval+1][MAXHMCOL+1];
double rdv[MaxSavedEval+1];
int SIV[MaxSavedEval+1];
int SI;
double MinTol = 0.005;
double draw[MXYZ+1], draw2[MXYZ+1], order2[MXYZ+1], Best[MXYZ+1], draw3[MXYZ+1],
order3[MXYZ+1], draw4[MXYZ+1], order4[MXYZ+1];
double BestProd[MAXHMRAW+1][MAXHMCOL+1];
double OLDOBJ, NEWOBJ;

int RUNS;
double OBJ;
int InRealType;   //Initial Realization from Sisim (=0) or from file (=1)
char InitFile[40]; //File with initial realization
int Subdomains;       //Using Subdomains, 0=No, 1=Yes
int RegionIndex[20]; //List of Indexes for Simulation Region/Subdomain
char RegionsFl[40];  //File with Region Indexes
int RegIndex[MXYZ+1];     //Vector storing the region indexes
int iReg;                 // Index of Region being perturbed
```

```
int VarType = 1;       // Modeling Distributions of Permeability (=0) or Rock Type/Multiphase
Functions (=1)
const int MaxRockTypes = 10;
double PermD[MaxRockTypes+1] = {0, 5.1, 43, 142, 390, 680, 0, 0, 0, 0, 0};
double PoroD[MaxRockTypes+1] = {0, 0.08, 0.13, 0.18, 0.22, 0.25, 0, 0, 0, 0, 0};

void swap (double& fel, double& sel);
int readparm();
int readparm2();
void  sortem (int lower, int upper, double array[], int Nother, double b[], double c[], double d[], double
e[][MXCUT+1], double f[], double g[], double h[]);
double acorni();
void ordrel();
void beyond(double cdfin[], double& zval, double& cdfval, double&ierr);
double powint(double xlow, double xhigh, double ylow, double yhigh, double xval, double power);
int locate (double xx[], int n, int is, int ie, double xv);
void sisim();
void setrot(double angle1, double angle2, double angle3, double anisotropy1, double anisotropy2, int
ind);
void setsupr(int nsec);
void getindx(int cells, double origin, double size, double locaction, int& foundIndex, bool& inflag);
void picksup(int irot);
double sqdist(double x1, double y1, double z1, double x2, double y2, double z2, int ind);
void ctable();
double cova3(double x1, double y1, double z1, double x2, double y2, double z2, int ivarg, int irot,
double& cmx);
void srchsuprsisim(double xloc, double yloc,double zloc, int irot, int &infoct);
int srchnd(int ix, int iy, int iz);
double krige(int ix, int iy, int iz, double xx, double yy, double zz, int icut, double gmean);
void ksol(int nright, int neq, int nsb,int& ising);

void updatedsisim();       // Create new realization considering the rd parameter.
void readProdHist();       // Read Real Production History
void readSimProd();            // Read Simulated Production History
void ProdMatchError(int sol);         // Estimation of Objective Function
void DekkerBrent();
void updateClass();

int k,j;
int gridindex;
double indexx;

//************************ MAIN PROGRAM ***************************************

int main()
{
    int run, i, j, outCounter, RN, NRegions;
    bool prtfirst;
    int RunFrom;
```

```
    prtfirst = false;
    RUNS = 0;
    ofstream rdout("RDout.txt", ios::trunc);
    rdout.close();
    ofstream peOut("PermOptResult.txt", ios::trunc);
    peOut.close();
    ofstream prOut("ProdOptResult.txt", ios::trunc);
    prOut.close();
    ofstream runOut("RunNumber.txt", ios::trunc);
    runOut.close();
//
//   Read the Parameter File and the Data:
//
    readparm();
    readProdHist();
    for (RN=1;RN<=Realizations;RN++)
    {
        RUNS = 0;
        for (i=1;i<=nxyz;i++)
        {
            draw[i] = acorni();
        }
//
// Call sisim for the simulation:
//
        sisim();
        RunFrom = 1;
        if ((RunFrom-1)==0)
        {
            system("$eclipse -file BASE");
            RUNS++;
            readSimProd();
            ProdMatchError(0);
        }
        else
        {
            objFunction[0] = 642.4;
        }
        OLDOBJ = objFunction[0];
        ofstream runOut1("RunNumber.txt", ios::ate);
        runOut1<<"Run #"<<RUNS<<" (initial) - OBJFunction: "<<objFunction[0]<<endl;
        runOut1.close();
        ofstream permOut1("PermOptResult.txt", ios::ate);
        if (RN!=1)
            permOut1<<endl<<endl;
        permOut1<<"Initial Perm Values for Realization "<<RN<<", ObjFunction:
"<<OLDOBJ<<endl<<1<<endl<<"Permeability";
        for (i=1;i<=nxyz;i++)
        {
            Best[i]=sim[i];
            permOut1<<endl<<sim[i];
```

```
        }
        permOut1.close();
        if ((RunFrom-1)==0)
        {
            ofstream prodOut1("ProdOptResult.txt", ios::ate);
            prodOut1<<endl<<endl<<"Initial Production for Realization "<<RN<<", ObjFunction:
"<<OLDOBJ<<endl;
            for (i=1;i<=numraw;i++)
            {
                for (j=1;j<=numcol;j++)
                {
                    prodOut1<<"  "<<SimProd[i][j];
                    BestProd[i][j] = SimProd[i][j];
                }
                prodOut1<<endl;
            }
            prodOut1<<endl;
            prodOut1.close();
        }
//
// Outer Loop
//
        for (outCounter = 1; outCounter <= maxIterOuterLoop; outCounter++)
        {
            if (Subdomains==0)
                NRegions = 1;
            else
                NRegions = Subdomains;
            for (iReg=1;iReg<=NRegions;iReg++) {
//
//   Work out a random path for this realization:
//
                for (i=1;i<=nxyz;i++)
                {
                    draw[i] = acorni();
                    draw2[i] = acorni();
                    draw3[i] = acorni();
                    draw4[i] = acorni();
                    order[i] = i;
                    order2[i] = i;
                    order3[i] = i;
                    order4[i] = i;
                }
                sortem(1,nxyz,draw,1,order,c,d,e,f,g,h);
                sortem(1,nxyz,draw2,1,order2,c,d,e,f,g,h);
                sortem(1,nxyz,draw3,1,order3,c,d,e,f,g,h);
                sortem(1,nxyz,draw4,1,order4,c,d,e,f,g,h);
//
// Save Path for inner loop
//
                for (i=1;i<=nxyz;i++)
```

```
                {
                    draw[i] = acorni();
                    draw2[i] = acorni();
                    draw3[i] = acorni();
                    draw4[i] = acorni();
                }
//
//   Initialize rd parameter
//
                for (run=1; run<=nSavedEval; run++)
                {
                    rd = rdspan[run];
//                  rdv[run] = (run-1)*(1.0/(nSavedEval-1));
//
//   External loop for Markov chain iterations
//
                    updatedsisim();
                    if (currentRDT==0.0)
                        objFunction[run] = OLDOBJ;
                    else
                    {
                        system("$eclipse -file BASE");
                        RUNS++;
                        readSimProd();
                        ProdMatchError(run);
                    }
                    ofstream prodOut2("ProdOptResult.txt", ios::ate);
                    prodOut2<<endl<<endl;
                    prodOut2<<"RD-SPANNING, SimRun# "<<RUNS<<", ObjFunction:
"<<objFunction[run]<<endl;
                    prodOut2<<"============================================="<<endl;
                    for (i=1;i<=numraw;i++)
                    {
                        for (j=1;j<=numcol;j++)
                        {
                            prodOut2<<"  "<<SimProd[i][j];
                        }
                        prodOut2<<endl;
                    }
                    prodOut2.close();
                    ofstream runOut2("RunNumber.txt", ios::ate);
                    runOut2<<"Run #"<<RUNS<<" (RD:"<<rd<<") - OBJFunction:
"<<objFunction[run]<<endl;
                    runOut2.close();
                    if (prtfirst) {
                        ofstream permOut("PermOptResult.txt", ios::ate);
                        permOut<<endl<<endl<<"Realization for RD =
"<<rd<<endl<<1<<endl<<"Perm";
                        for (i=1;i<=nxyz;i++)
                        {
                            permOut<<endl<<sim[i];
```

```
                    }
                    permOut.close();
                }
            }
            prtfirst = false;
            ofstream rdout("RDout.txt", ios::ate);
            rdout<<endl<<endl<<"OutLoop# "<<outCounter<<", Region# "<<iReg<<", SimRun#
"<<RUNS<<endl;
            rdout<<"================================================"<<endl;
            rdout<<"  Initial "<<nSavedEval<<" RD Evaluations"<<endl;
            for (run=1;run<=nSavedEval;run++) {
                rdv[run] = rdspan[run];
//              rdv[run] = (run-1)*(1.0/(nSavedEval-1));
                rdout<<"     RD: "<<rdv[run]<<":   ObjFunction: "<<objFunction[run]<<endl;
            }
            rdout<<"  DekkerBrent's "<<maxIterInnerLoop<<" RD Evaluations"<<endl;
            rdout.close();
            DekkerBrent();
            if (NEWOBJ<OLDOBJ)
            {
                OLDOBJ = NEWOBJ;
                updateClass();
            ofstream permOut("PermOptResult.txt", ios::ate);
                permOut<<endl<<endl<<"OutLoop# "<<outCounter<<", Region# "<<iReg;
                permOut<<", SimRun# "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
                permOut<<"================================================"<<endl;
                permOut<<"Permeability"<<endl;
                for (i=1;i<=nxyz;i++)
                {
                    permOut<<sim[i]<<endl;
                    Best[i] = sim[i];
                }
                permOut.close();
                ofstream prodOut("ProdOptResult.txt", ios::ate);
                prodOut<<endl<<endl<<"OutLoop# "<<outCounter<<", Region# "<<iReg;
                prodOut<<", SimRun# "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
                prodOut<<"================================================"<<endl;
                for (i=1;i<=numraw;i++)
                {
                    for (j=1;j<=numcol;j++)
                    {
                        prodOut<<"  "<<SimProd[i][j];
                        BestProd[i][j] = SimProd[i][j];
                    }
                    prodOut<<endl;
                }
                prodOut.close();
            }
        }
        if (Subdomains>0) {
            for (i=1;i<=nxyz;i++)
```

```
                {
                    draw[i] = acorni();
                    order[i] = i;
                }
                sortem(1,nxyz,draw,1,order,c,d,e,f,g,h);
//
// Save Path for inner loop
//
                for (i=1;i<=nxyz;i++)
                {
                    draw[i] = acorni();
                }
                updatedsisim();
                updateClass();
                system("$eclipse -file BASE");
                RUNS++;
                readSimProd();
                ProdMatchError(0);
                NEWOBJ = objFunction[0];
                OLDOBJ = NEWOBJ;
                updateClass();
            ofstream permOut("PermOptResult.txt", ios::ate);
                permOut<<endl<<endl<<"OutLoop# "<<outCounter<<", AllRegions";
                permOut<<", SimRun# "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
                permOut<<"================================================="<<endl;
                permOut<<"Permeability"<<endl;
                for (i=1;i<=nxyz;i++)
                {
                    permOut<<sim[i]<<endl;
                    Best[i] = sim[i];
                }
                permOut.close();
                ofstream prodOut("ProdOptResult.txt", ios::ate);
                prodOut<<endl<<endl<<"OutLoop# "<<outCounter<<", AllRegions";
                prodOut<<", SimRun# "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
                prodOut<<"================================================="<<endl;
                for (i=1;i<=numraw;i++)
                {
                    for (j=1;j<=numcol;j++)
                    {
                        prodOut<<"  "<<SimProd[i][j];
                        BestProd[i][j] = SimProd[i][j];
                    }
                    prodOut<<endl;
                }
                prodOut.close();
            }
//
//End OUTER LOOP
//
        }
```

```
        ofstream permOut("PermOptResult.txt", ios::ate);
        permOut<<endl<<endl<<"Final Realization "<<RN;
        permOut<<", TotalRuns: "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
        permOut<<"=========================================================="<<endl;
        permOut<<"Best Permeability"<<endl;
        for (i=1;i<=nxyz;i++)
        {
            permOut<<Best[i]<<endl;
        }
        permOut.close();
        ofstream prodOut("ProdOptResult.txt", ios::ate);
        prodOut<<endl<<endl<<"Final Production, Realization: "<<RN;
        prodOut<<", TotalRuns: "<<RUNS<<", ObjFunction: "<<OLDOBJ<<endl;
        prodOut<<"=========================================================="<<endl;
        for (i=1;i<=numraw;i++)
        {
            for (j=1;j<=numcol;j++)
            {
                prodOut<<"  "<<BestProd[i][j];
            }
            prodOut<<endl;
        }
        prodOut.close();
    }
//
//  Finished:
//
    return 0;
}


// ****************** READ PARAMETER FILE ************************************

int readparm()
{
    int index, i, j, k, ind, nvari, ic, icut, istart1, istarti, ist, index1, indexi;
    double temp, av, ss, vrt, xd, tcdf, oldcp, cp, xx, yy, zz, test, cdfval, zval, clos, pppp;
    char tempch[80];
    char descr[50];
//
//  Read Input Parameters:
//
    ifstream fin1("hmissim.par", ios::nocreate);
    if (!fin1)
    {
        cout<<"There is no Parameter file"<<endl;
        exit(1);
    }
    ofstream fout;
    fout.open("ReadParameter.txt");
    fin1>>descr>>ivtype;
```

```
fout<<"Variable Type (1=continuous, 0=categorical)= "<<ivtype<<endl;
fin1>>descr>>ncut;
fout<<"Number of Thresholds / Categories = "<<ncut<<endl;
if(ncut>MAXCUT)
{
    cout<<"ncut is too big - modify MAXCUT";
     exit(1);
}
fin1>>descr;
fout<<"Thresholds / Categories = "<<endl;
for (i=1;i<=ncut;i++)
{
    fin1>>thres[i];
    fout<<thres[i]<<"  ";
}
fout<<endl;
fin1>>descr;
fout<<"Global CDF / PDF = "<<endl;
for (i=1;i<=ncut;i++)
{
    fin1>>cdf[i];
    fout<<cdf[i]<<"  ";
}
fout<<endl;
fin1>>descr>>datafl;
fout<<"Data File = "<<datafl<<endl;
fin1>>descr>>ixl>>iyl>>izl>>ivrl;
fout<<"Input columns = "<<ixl<<"  "<<iyl<<"  "<<izl<<"  "<<ivrl<<endl;
fin1>>descr>>softfl;
fout<<"Soft Data File = "<<softfl<<endl;
fin1>>descr>>ixs>>iys>>izs;
fout<<"Columns = "<<ixs<<"  "<<iys<<"  "<<izs<<endl;
fout<<"Indicators = "<<endl;
for (i=1;i<=ncut;i++)
{
    fin1>>ivrs[i];
    fout<<ivrs[i]<<"  ";
}
fout<<endl;
fin1>>descr>>imbsim;
fout<<"Markov-Bayes Simulation = "<<imbsim<<endl;
fin1>>descr;
if (imbsim==1)
{
    for (i=1;i<=ncut;i++)
        fin1>>beez[i];
}
else
{
    for (i=1;i<=ncut;i++)
        fin1>>temp;
```

```
}
fin1>>descr>>tmin>>tmax;
fout<<"Trimming Limits = "<<tmin<<"  "<<tmax<<endl;
fin1>>descr>>zmin>>zmax;
fout<<"Data Limits (Tails) = "<<zmin<<"  "<<zmax<<endl;
fin1>>descr>>ltail>>ltpar;
fout<<"Lower Tail = "<<ltail<<"  "<<ltpar<<endl;
fin1>>descr>>middle>>mpar;
fout<<"Middle = "<<middle<<"  "<<mpar<<endl;
fin1>>descr>>utail>>utpar;
fout<<"Upper Tail = "<<utail<<"  "<<utpar<<endl;
fin1>>descr>>tabfl;
fout<<"File for Tab. Quant. = "<<tabfl<<endl;
fin1>>descr>>itabvr>>itabwt;
fout<<"Columns for Variable & Weight = "<<itabvr<<"  "<<itabwt<<endl;
fin1>>descr>>idbg;
fout<<"Debugging Level = "<<idbg<<endl;
fin1>>descr>>dbgfl;
fout<<"Debugging File = "<<dbgfl<<endl;
fin1>>descr>>outfl;
fout<<"Output File = "<<outfl<<endl;
ofstream fout1(outfl, ios::trunc);
fout1.close();
fin1>>descr>>PHFile;
fout<<"Production History File = "<<PHFile<<endl;
fin1>>descr>>nx>>xmn>>xsiz;
fout<<"X Grid Specification = "<<nx<<"  "<<xmn<<"  "<<xsiz<<endl;
fin1>>descr>>ny>>ymn>>ysiz;
fout<<"Y Grid Specification = "<<ny<<"  "<<ymn<<"  "<<ysiz<<endl;
fin1>>descr>>nz>>zmn>>zsiz;
fout<<"Z Grid Specification = "<<nz<<"  "<<zmn<<"  "<<zsiz<<endl;
nxy  = nx*ny;
nxyz = nx*ny*nz;
fin1>>descr>>ixv[1];
fout<<"Random Number Seed = "<<ixv[1]<<endl;
for (i=1;i<=1000;i++)
    pppp = acorni();
fin1>>descr>>ndmax;
fout<<"Max. Orig. Data for Krig. = "<<ndmax<<endl;
fin1>>descr>>nodmax;
fout<<"Max Prev Sim Nodes = "<<nodmax<<endl;
fin1>>descr>>maxsec;
fout<<"Max Soft Indicator Data = "<<maxsec<<endl;
fin1>>descr>>sstrat;
fout<<"Search Strategy = "<<sstrat<<endl;
fin1>>descr>>mults>>nmult;
fout<<"Multiple Grid Search Flag = "<<mults<<"  "<<nmult<<endl;
fin1>>descr>>noct;
fout<<"Max Per Octant = "<<noct<<endl;
fin1>>descr>>radius>>radius1>>radius2;
fout<<"Search Radii = "<<radius<<"  "<<radius1<<"  "<<radius2<<endl;
```

```
if(radius<EPSLON)
{
     cout<<"Radius must be greater than zero"<<endl;
      exit(1);
}
radsqd = radius*radius;
sanis1 = radius1 / radius;
sanis2 = radius2 / radius;
fin1>>descr>>sang1>>sang2>>sang3;
fout<<"Search Anisotropy Angles = "<<sang1<<"  "<<sang2<<"  "<<sang3<<endl;
fin1>>descr>>mik>>cutmik;
fout<<"Median IK Switch = "<<mik<<"  "<<cutmik<<endl;
fin1>>descr>>ktype;
fout<<"Kriging Type Switch = "<<ktype<<endl;
fin1>>descr>>InRealType;
fin1>>descr>>InitFile;
if (InRealType==0)
     fout<<"Initial Realization Calculated by sisim"<<endl;
else {
     fout<<"Initial Realization from file"<<endl;
     fout<<"File with Initial Realization : "<<InitFile<<endl;
}
fin1>>descr>>Subdomains;
for (i=1;i<=Subdomains;i++)
     fin1>>RegionIndex[i];
fin1>>descr>>RegionsFl;
if (Subdomains==0)
     fout<<"Simulation With No Subdomains"<<endl;
else
{
     fout<<"Simulation for Subdomain Index = ";
     for (i=1;i<=Subdomains;i++)
          fout<<RegionIndex[i]<<" ";
     fout<<endl;
     fout<<"File with Regions: "<<RegionsFl<<endl;
     ifstream RegIn(RegionsFl, ios::nocreate);
     if (!RegIn)
     {
          cout<<"There is no Regions file"<<endl;
          exit(1);
     }
     for (k=nz;k>=1;k--)
     {
          for(j=ny;j>=1;j--)
          {
               for (i=1;i<=nx;i++)
               {
                    ind = (k-1)*nxy + (j-1)*nx + i;
                    RegIn>>RegIndex[ind];
               }
          }
```

```
        }
        RegIn.close();
    }
//
//  Perform some quick error checking:
//
    if(nx>MAXX)
    {
        cout<<"nx is too big - modify MAXX"<<endl;
         exit(1);
    }
    if(ny>MAXY)
    {
        cout<<"ny is too big - modify MAXY"<<endl;
         exit(1);
    }
    if(nz>MAXZ)
    {
        cout<<"nz is too big - modify MAXZ"<<endl;
         exit(1);
    }
    fin1>>descr>>nSavedEval;
    fout<<"Number of initial RD Evaluations: "<<nSavedEval<<endl;
    fin1>>descr>>maxIterInnerLoop;
    fout<<"Number of inner loop iterations: "<<maxIterInnerLoop<<endl;
    fin1>>descr>>maxIterOuterLoop;
    fout<<"Number of outer loop iterations: "<<maxIterOuterLoop<<endl;
    fout<<"Variogram Structure = "<<endl;
    for (i=1;i<=ncut;i++)
    {
        fin1>>descr;
        fin1>>nst[i]>>c0[i];
        fout<<nst[i]<<"  "<<c0[i]<<endl;
        istart = 1 + (i-1)*MAXNST;
        for (j=1;j<=nst[i];j++)
        {
            index = istart + j - 1;
            fin1>>it[index]>>cc[index]>>ang1[index]>>ang2[index]>>ang3[index];
            fout<<it[index]<<"  "<<cc[index]<<"  "<<ang1[index]<<"  "<<ang2[index]<<"
"<<ang3[index]<<endl;
            if(it[index]==3)
            {
                cout<<"Gaussian Model Not Allowed!"<<endl;
                 exit(1);
            }
            fin1>>aa[index]>>aa1>>aa2;
            fout<<aa[index]<<"  "<<aa1<<"  "<<aa2<<endl;
            anis1[index] = aa1 / __max(EPSLON,aa[index]);
            anis2[index] = aa2 / __max(EPSLON,aa[index]);
        }
    }
```

```
    fin1.close();
    fout.close();
//
//  Check to make sure the data file exists, then either read in the
//  data or write a warning:
//
    ifstream fin2(datafl, ios::nocreate);
    if (!fin2)
    {
        cout<<"There is no Data File"<<endl;
        exit(1);
    }
//
//  The data file exists so open the file and read in the header
//  information.
//
    ofstream fout2;
    fout2.open("ReadData.txt");
    nd = 0;
    av = 0.0;
    ss = 0.0;
    fin2.getline(title,80);
    fout2<<title<<endl;
    fin2>>nvari;
    fout2<<"Number of Columns = "<<nvari<<endl;
    for (i=1;i<=(nvari+1);i++)
    {
        fin2.getline(tempch,80);
        fout2<<tempch<<endl;
    }
//
//  Read all the data until the end of the file:
//
    while (!fin2.eof())
    {
        for (i=1;i<=nvari;i++)
        {
            fin2>>var[i];
            fout2<<var[i]<<"  ";
        }
        fout2<<endl;
        vrt = var[ivrl];
        if ((vrt>=tmin)&&(vrt<=tmax))
        {
            nd += 1;
            if (nd>MAXDAT)
            {
                cout<<"ERROR exceeded MAXDAT";
                exit(1);
            }
         x[nd] = xmn;
```

```
            y[nd] = ymn;
        z[nd] = zmn;
        if (ixl>0)
            x[nd] = var[ixl];
          if (iyl>0)
            y[nd] = var[iyl];
        if (izl>0)
            z[nd] = var[izl];
          av += vrt;
        ss += vrt*vrt;
//
//   The indicator data are constructed knowing the thresholds and the
//   data value.
//
        if (ivtype==0)
          {
            for (ic=1;ic<=ncut;ic++)
            {
              vr[nd][ic] = 0.0;
           if (int(vrt+0.5)==int(thres[ic]+0.5))
                vr[nd][ic] = 1.0;
            }
          }
        else
          {
            for (ic=1;ic<=ncut;ic++)
            {
              vr[nd][ic] = 1.0;
              if (vrt>thres[ic])
                vr[nd][ic] = 0.0;
            }
          }
          vr[nd][MXCUT] = vrt;
        }
    }
    fin2.close();
//
//   Compute the averages and variances as an error check for the user:
//
    xd = __max((1.0*nd),1.0);
    av = av/xd;
    ss = (ss/xd) - av*av;
    fout2<<endl;
    fout2<<"Variable Number = "<<ivrl<<endl;
    fout2<<"Nuber of Acceptable Data = "<<nd<<endl;
    fout2<<"Equal Weighted Average = "<<av<<endl;
    fout2<<"Equal Weighted Variance = "<<ss<<endl;
//
//   Check to make sure that the grid is compatible with the data:
//
    if ((ixl<=0)&&(nx>1))
```

```
    {
        fout2<<"ERROR there is no X coordinate in data file";
        exit(1);
    }
    if ((iyl<=0)&&(ny>1))
    {
        fout2<<"ERROR there is no Y coordinate in data file";
        exit(1);
    }
    if ((izl<=0)&&(nz>1))
    {
        fout2<<"ERROR there is no Z coordinate in data file";
        exit(1);
    }
    fout2.close();
//
//  Now, if required, read in the tabulated values for details of the dist
//
    if ((ltail==3)||(middle==3)||(utail==3))
    {
        ifstream fin3(tabfl, ios::nocreate);
        if (!fin3)
        {
            cout<<"There is no File with Tabulated Quant."<<endl;
            cout<<"ERROR on Extrapolation Option"<<endl;
            exit(1);
        }
        fin3.getline(tempch,80);
        fin3>>nvari;
        for (i=1;i<=(nvari+1);i++)
            fin3.getline(tempch,80);
        tcdf = 0.0;
        ng   = 0;
        while (!fin3.eof())
        {
            for (i=1;i<=nvari;i++)
                fin3>>var[i];
            if ((var[itabvr]>=tmin)&&(var[itabvr]<=tmax))
            {
                ng += 1;
                if (ng>MAXTAB)
                {
                    cout<<"ERROR exceeded MAXTAB"<<endl;
                    exit(1);
                }
                gcut[ng] = var[itabvr];
                gcdf[ng] = 1.0;
                if (itabwt>0)
                    gcdf[ng] = var[itabwt];
                tcdf += gcdf[ng];
            }
```

```
        }
        fin3.close();
//
//   Sort in ascending order and keep track of where the tabulated values
//   switch classes:
//
        if (tcdf<=0.0)
        {
            cout<<"ERROR: either the weights are zero or there are no tabulated data"<<endl;
            exit(1);

        }
        sortem (1,ng,gcut,1,gcdf,c,d,e,f,g,h);
//
//   Set up gcdf for tabulated quantiles:
//
        oldcp = 0.0;
        cp = 0.0;
        tcdf = 1.0/tcdf;
        for (i=1;i<=ng;i++)
        {
            cp += gcdf[i]*tcdf;
            gcdf[i] = (cp + oldcp)*0.5;
            oldcp = cp;
        }
    }
//
//   Direct input of indicator data:
//
    nhd = nd;
    ifstream fin4(softfl, ios::nocreate);
    if (!fin4)
    {
        cout<<"There is no Soft Data File"<<endl;
    }
    if (fin4)
    {
        fin4.getline(tempch,80);
        fin4>>nvari;
        if (ivrs[ncut]>nvari)
        {
            cout<<"ERROR: too few variables in softfl - inconsistent with parameters"<<endl;
            exit(1);
        }
        for (i=1;i<=(nvari+1);i++)
            fin4.getline(tempch,80);
        while (!fin4.eof())
        {
            for (j=1;j<=nvari;j++)
                fin4>>var[j];
//
```

```
//   Don't keep soft data co-located with hard data:
//
            xx = xmn;
            yy = ymn;
            zz = zmn;
            if (ixs>0)
            xx = var[ixs];
            if (iys>0)
                yy = var[iys];
            if (izs>0)
                zz = var[izs];
            for (i=1;i<=nhd;i++)
                test = fabs(xx - x[i]) + fabs(yy - y[i]) + fabs(zz - z[i]);
            if (test>EPSLON)
            {
//
//   Accept this data:
//
                nd++;
                if (nd>MAXDAT)
                {
                    cout<<"ERROR: Exceeded memory for data - softfl";
                    exit(1);
                }
                x[nd] = xx;
                y[nd] = yy;
                z[nd] = zz;
                for (j=1;j<=ncut;j++)
                {
                    i = ivrs[j];
                    vr[nd][j] = var[i];
                    ccdf[j] = var[i];
                }
//
//   Draw a value for this soft distribution (in case the distribution is
//   co-located with a grid node and Markov-Bayes is not used):
//
                cdfval = acorni();
                ordrel();
                zval = UNEST;
                beyond(ccdfo,zval,cdfval,ierr);
                vr[nd][MXCUT] = zval;
//
//    If performing median IK then check for missing values:
//
                if (mik==1)
                {
                    for (ic=1;ic<=ncut;ic++)
                    {
                        if (vr[nd][ic]<0.0)
                        {
```

```
                                cout<<"Missing values on Median IK";
                                exit(1);
                            }
                        }
                    }
                }
            }
            fin4.close();
        }
//
//  Load the right variogram as the first one if performing median IK:
//
        if (mik==1)
        {
            icut = 1;
            clos = fabs(cutmik - thres[1]);
            for (ic=2;ic<=ncut;ic++)
            {
                test = fabs(cutmik - thres[ic]);
                if (test<clos)
                {
                    icut = ic;
                    clos = test;
                }
            }
            c0[1] = c0[icut];
            nst[1] = nst[icut];
            istart1 = 1;
            istarti = 1 + (icut-1)*MAXNST;
            for (ist=1;ist<=nst[1];ist++)
            {
                index1 = istart1 + ist - 1;
                indexi = istarti + ist - 1;
                it[index1] = it[indexi];
                aa[index1] = aa[indexi];
                cc[index1] = cc[indexi];
                ang1[index1] = ang1[indexi];
                ang2[index1] = ang2[indexi];
                ang3[index1] = ang3[indexi];
                anis1[index1] = anis1[indexi];
                anis2[index1] = anis2[indexi];
            }
        }
//
//  Open the output file and return:
//
        ofstream fout3;
        fout3.open(outfl);
        fout3<<title<<endl;
        fout3<<"1"<<endl;
        fout3<<"Simulated Value"<<endl;
```

```
    fout3.close();
    return 0;
}


//********************** SORTING ARRAYS ************************************


void  sortem (int lower, int upper, double array[], int Nother, double b[], double c[], double d[], double
e[][MXCUT+1], double f[], double g[], double h[])
{
    int p, q, r, i;
    double pivot;
    q = lower;
    r = upper;
    p = (lower + upper) / 2;
    pivot = array[p];
    do
    {
        while (array[q] < pivot)
            q++;
        while (array[r] > pivot)
            r--;
        if (q <= r)
        {
            swap(array[q], array[r]);
            if (Nother>=1)
            {
                swap(b[q], b[r]);
                if (Nother>=2)
                {
                    swap(c[q], c[r]);
                    if (Nother>=3)
                    {
                        swap(d[q], d[r]);
                        if (Nother>=4)
                        {
                            for (i=1;i<=MXCUT;i++)
                                swap(e[q][i], e[r][i]);
                            if (Nother>=5)
                            {
                                swap(f[q], f[r]);
                                if (Nother>=6)
                                {
                                    swap(g[q], g[r]);
                                    if (Nother>=7)
                                        swap(h[q], h[r]);

                                }
                            }
                        }
```

```
                    }
                }
            }
            q++;
            r--;
        }
    }
    while (q < r);
    if (r > lower) sortem (lower, r, array, Nother,b,c,d,e,f,g,h);
    if (q < upper) sortem (q, upper, array, Nother,b,c,d,e,f,g,h);
}


//********************************** SWAP VALUES ************************


void swap (double &fel, double &sel)
{
    double temp;
    temp=fel;
    fel=sel;
    sel=temp;
    return;
}


//************************** RANDOM NUMBER GENERATOR *********************


double acorni()
{
    int i;
    double acor;
    for (i=1;i<=KORDEI; i++)
    {
        ixv[i+1] = ixv[i+1] + ixv[i];
        if (ixv[i+1]>=MAXINT)
            ixv[i+1] -= MAXINT;
    }
    acor = double(ixv[KORDEI+1])/MAXINT;
    return acor;
}

//******************** CORRECT ORDER RELATION ON CDF ********************


void ordrel()
{
    int i;
    double ccdf1[MAXCUT+1], ccdf2[MAXCUT+1], sumcdf, viol;
//
```

```
// Make sure there is enough temporary storage:
//
    if (ncut>MAXCUT)
    {
        cout<<"MAXCUT<ncut - There is not enough temporary storage allocated";
        exit(1);
    }
//
// Make sure conditional cdf is within [0,1]:
//
    for (i=1;i<=ncut;i++)
    {
        if (ccdf[i]<0.0)
        {
            ccdf1[i] = 0.0;
            ccdf2[i] = 0.0;
        }
        else
        {
            if (ccdf[i]>1.0)
            {
                ccdf1[i] = 1.0;
                ccdf2[i] = 1.0;
            }
            else
            {
                ccdf1[i] = ccdf[i];
                ccdf2[i] = ccdf[i];
            }
        }
    }
//
// Correct sequentially up, then down, and then average:
//
    if (ivtype==0)
    {
        sumcdf = 0.0;
        for (i=1;i<=ncut;i++)
            sumcdf += ccdf1[i];
        if (sumcdf<=0.0)
            sumcdf = 1.0;
        for (i=1;i<=ncut;i++)
            ccdfo[i] = ccdf1[i]/sumcdf;
    }
    else
    {
        for (i=2;i<=ncut;i++)
        {
            if (ccdf1[i]<ccdf1[i-1])
                ccdf1[i] = ccdf1[i-1];
        }
```

```
        for (i=ncut-1;i>=1;i--)
        {
            if (ccdf2[i]>ccdf2[i+1])
                ccdf2[i] = ccdf2[i+1];
        }
        for (i=1;i<=ncut;i++)
            ccdfo[i] = 0.5*(ccdf1[i]+ccdf2[i]);
    }
//
// Accumulate error statistics:
//
    for (i=1;i<=ncut;i++)
    {
        if (ccdf[i]!=ccdfo[i])
        {
            viol = fabs(ccdf[i]-ccdfo[i]);
            nviol[i] += 1;
            aviol[i] += viol;
            xviol[i] = __max(xviol[i],viol);
        }
    }
//
// Return with corrected CDF:
//
    return;
}


//********************** SAMPLE FROM DISCRETE CDF ***************************


void beyond(double cdfin[], double& zval, double& cdfval, double&ierr)
{
    double lambda, cum, powr, temp;
    int i, cclow, cchigh, ipart, idat, iupp, ilow;
//
// Check for both "zval" and "cdfval" defined or undefined:
//
    ierr = 1;
    if ((zval>UNEST)&&(cdfval>UNEST))
        return;
    if ((zval<=UNEST)&&(cdfval<=UNEST))
        return;
//
// Handle the case of a categorical variable:
//
    if (ivtype==0)
    {
        cum = 0;
        for (i=1;i<=ncut;i++)
        {
```

```
            cum += cdfin[i];
            if (cdfval<=cum)
            {
                zval = thres[i];
                return;
            }
        }
        return;
    }
//
// Figure out what part of distribution: ipart = 0 - lower tail
//                                       ipart = 1 - middle
//                                       ipart = 2 - upper tail
    ierr = 0;
    ipart = 1;
    if (zval>UNEST)
    {
        if (zval<=thres[1])
            ipart = 0;
        if (zval>=thres[ncut])
            ipart = 2;
    }
    else
    {
        if (cdfval<=cdfin[1])
            ipart = 0;
        if (cdfval>=cdfin[ncut])
            ipart = 2;
    }
//
// ARE WE IN THE LOWER TAIL?
//
    if (ipart==0)
    {
        if (ltail==1)
        {
//
// Straight Linear Interpolation:
//
            powr = 1.0;
            if (zval>UNEST)
                cdfval = powint(zmin,thres[1],0.0,cdfin[1],zval,powr);
            else
                zval = powint(0.0,cdfin[1],zmin,thres[1],cdfval,powr);
        }
        else
        {
            if (ltail==2)
            {
//
// Power Model interpolation to lower limit "zmin"?
```

```
//
                if (zval>UNEST)
                    cdfval = powint(zmin,thres[1],0.0,cdfin[1],zval,ltpar);
                else
                {
                    powr = 1.0/ltpar;
                    zval = powint(0.0,cdfin[1],zmin,thres[1],cdfval,powr);
                }
//
// Linear interpolation between the rescaled global cdf?
//
            }
        else
            {
                if (ltail==3)
                {
                    if (zval>UNEST)
                    {
//
// Computing the cdf value. Locate the point and the class bound:
//
                        idat = locate(gcut,ng,1,ng,zval);
                        iupp = locate(gcut,ng,1,ng,thres[1]);
//
// Straight linear interpolation if no data; otherwise, linear:
//
                        if ((idat<=0)||(idat>=ng)||(iupp<=0)||(iupp>=ng))
                            cdfval = powint(zmin,gcut[1],0.0,gcdf[1],zval,1.);
                        else
                        {
                            temp = powint(gcut[idat],gcut[idat+1],gcdf[idat],gcdf[idat+1],zval,1.);
                            cdfval = temp*cdfin[1]/gcdf[iupp];
                        }
                    }
                    else
                    {
//
// Computing Z value: Are there any data out in the tail?
//
                        iupp = locate(gcut,ng,1,ng,thres[1]);
//
// Straight linear interpolation if no data; otherwise, local linear
// interpolation:
//
                        if ((iupp<=0)||(iupp>=ng))
                            zval = powint(0.0,gcdf[1],zmin,gcut[1],cdfval,1.);
                        else
                        {
                            temp = cdfval*gcdf[iupp]/cdfin[1];
                            idat = locate(gcdf,ng,1,ng,temp);
                            if ((idat<=0)||(idat>=ng))
```

```
                                zval = powint(0.0,gcdf[1],zmin,gcut[1],cdfval,1.);
                            else
                                zval = powint(gcdf[idat],gcdf[idat+1],gcut[idat],gcut[idat+1],temp,1.);
                        }
                    }
                }
                else
                {
//
// Error situation - unacceptable option:
//
                    ierr = 2;
                    return;
                }
            }
        }
    }
//
//FINISHED THE LOWER TAIL,  ARE WE IN THE MIDDLE?
//
    if (ipart==1)
    {
//
// Establish the lower and upper limits:
//
        if (zval>UNEST)
            cclow = locate(thres,ncut,1,ncut,zval);
        else
            cclow = locate(cdfin,ncut,1,ncut,cdfval);
        cchigh = cclow + 1;
        if (middle==1)
        {
//
// Straight Linear Interpolation:
//
            powr = 1.0;
            if (zval>UNEST)
                cdfval = powint(thres[cclow],thres[cchigh],cdfin[cclow],cdfin[cchigh],zval,powr);
            else
                zval = powint(cdfin[cclow],cdfin[cchigh],thres[cclow],thres[cchigh],cdfval,powr);
//
// Power interpolation between class bounds?
//
        }
        else
        {
            if (middle==2)
            {
                if (zval>UNEST)
                    cdfval = powint(thres[cclow],thres[cchigh],cdfin[cclow],cdfin[cchigh],zval,mpar);
                else
```

```
                {
                    powr = 1.0/mpar;
                    zval = powint(cdfin[cclow],cdfin[cchigh],thres[cclow],thres[cchigh],cdfval,powr);
                }
            }
//
// Linear interpolation between the rescaled global cdf?
//
            else
            {
                if (middle==3)
                {
                    ilow = locate(gcut,ng,1,ng,thres[cclow]);
                    iupp = locate(gcut,ng,1,ng,thres[cchigh]);
                    if (gcut[ilow]<thres[cclow])
                        ilow++;
                    if (gcut[iupp]>thres[cchigh])
                        iupp--;
                    if (zval>UNEST)
                    {
                        idat = locate(gcut,ng,1,ng,zval);
//
// Straight linear interpolation if no data; otherwise, local linear
// interpolation:
//
                        if
((idat<=0)||(idat>=ng)||(ilow<=0)||(ilow>=ng)||(iupp<=0)||(iupp>=ng)||(iupp<=ilow))
                            cdfval =
powint(thres[cclow],thres[cchigh],cdfin[cclow],cdfin[cchigh],zval,1.);
                        else
                        {
                            temp = powint(gcut[idat],gcut[idat+1],gcdf[idat],gcdf[idat+1],zval,1.);
                            cdfval = powint(gcdf[ilow],gcdf[iupp],cdfin[cclow],cdfin[cchigh],temp,1.);
                        }
                    }
                    else
                    {
//
// Straight linear interpolation if no data; otherwise, local linear
// interpolation:
//
                        if ((ilow<=0)||(ilow>=ng)||(iupp<=0)||(iupp>=ng)||(iupp<=ilow))
                            zval =
powint(cdfin[cclow],cdfin[cchigh],thres[cclow],thres[cchigh],cdfval,1.);
                        else
                        {
                            temp = powint(cdfin[cclow],cdfin[cchigh],gcdf[ilow],gcdf[iupp],cdfval,1.);
                            idat = locate(gcdf,ng,1,ng,temp);
                            if (gcut[idat]<thres[cclow])
                                idat++;
                            if ((idat<=0)||(idat>=ng)||(gcut[idat+1]>thres[cchigh]))
```

```
                                zval =
powint(cdfin[cclow],cdfin[cchigh],thres[cclow],thres[cchigh],cdfval,1.);
                            else
                                zval = powint(gcdf[idat],gcdf[idat+1],gcut[idat],gcut[idat+1],temp,1.);
//ERROR?????????            zval = powint(gcdf[idat],gcdf[idat+1],gcut[idat],gcut[idat+1],temp,1.);
                        }
                    }
                }
                else
                {
//
// Error situation - unacceptable option:
//
                    ierr = 2;
                    return;
                }
            }
        }
    }
//
// FINISHED THE MIDDLE,  ARE WE IN THE UPPER TAIL?
//
    if (ipart==2)
    {
        if (utail==1)
        {
            powr = 1.0;
            if (zval>UNEST)
                cdfval = powint(thres[ncut],zmax,cdfin[ncut],1.0,zval,powr);
            else
                zval = powint(cdfin[ncut],1.0,thres[ncut],zmax,cdfval,powr);
        }
        else
        {
            if (utail==2)
            {
//
// Power interpolation to upper limit "utpar"?
//
                if (zval>UNEST)
                    cdfval = powint(thres[ncut],zmax,cdfin[ncut],1.0,zval,utpar);
                else
                {
                    powr = 1.0/utpar;
                    zval = powint(cdfin[ncut],1.0,thres[ncut],zmax,cdfval,powr);
                }
            }
//
// Linear interpolation between the rescaled global gcdf?
//
            else
```

```
            {
                if (utail==3)
                {
                    if (zval>UNEST)
                    {
//
// Approximately Locate the point and the class bound:
//
                        idat = locate(gcut,ng,1,ng,zval);
                        ilow = locate(gcut,ng,1,ng,thres[ncut]);
                        if (gcut[idat]<zval)
                            idat++;
                        if (gcut[ilow]<thres[ncut])
                            ilow++;
//
// Straight linear interpolation if no data; otherwise, local linear
// interpolation:
//
                        if ((idat<=0)||(idat>=ng)||(ilow<=0)||(ilow>=ng))
                            cdfval = powint(thres[ncut],zmax,cdfin[ncut],1.0,zval,1.);
                        else
                        {
                            temp = powint(gcut[idat],gcut[idat+1],gcdf[idat],gcdf[idat+1],zval,1.);
                            cdfval = powint(gcdf[ilow],1.0,cdfin[ncut],1.0,temp,1.);
                        }
                    }
                    else
                    {
//
// Computing Z value: Are there any data out in the tail?
//
                        ilow = locate(gcut,ng,1,ng,thres[ncut]);
                        if (gcut[ilow]<thres[ncut])
                            ilow++;
//
// Straight linear interpolation if no data; otherwise, local linear
// interpolation:
//
                        if ((ilow<=0)||(ilow>=ng))
                            zval = powint(cdfin[ncut],1.0,thres[ncut],zmax,cdfval,1.);
                        else
                        {
                            temp = powint(cdfin[ncut],1.0,gcdf[ilow],1.0,cdfval,1.);
                            idat = locate(gcdf,ng,1,ng,temp);
                            if (gcut[idat]<thres[ncut])
                                idat++;
                            if (idat>=ng)
                                zval = powint(cdfin[ncut],1.0,thres[ncut],zmax,cdfval,1.);
                            else
                                zval = powint(gcdf[idat],gcdf[idat+1],gcut[idat],gcut[idat+1],temp,1.);
                        }
```

```
                    }
                }
//
// Fit a Hyperbolic Distribution?
//
                else
                {
                    if (utail==4)
                    {
//
// Figure out "lambda" and required info:
//
                        lambda = pow(thres[ncut],utpar)*(1.0 - cdfin[ncut]);
                        if (zval>UNEST)
                            cdfval = 1.0 - (lambda/pow(zval,utpar));
                        else
                            zval = pow((lambda/(1.0-cdfval)),(1.0/utpar));
                    }
                    else
                    {
//
// Error situation - unacceptable option:
//
                        ierr = 2;
                        return;
                    }
                }
            }
        }
    }
    if (zval<zmin)
        zval = zmin;
    if (zval>zmax)
        zval = zmax;
//
// All finished - return:
//
    return;
}


//**************************** POWER INTERPOLATION ***************************


double powint(double xlow, double xhigh, double ylow, double yhigh, double xval, double power)
{
    double pwit;
    if ((xhigh - xlow)<EPSLON)
        pwit = (yhigh + ylow)/2.0;
    else
        pwit = ylow + (yhigh - ylow)*pow(((xval-xlow)/(xhigh-xlow)),power);
```

```
    return pwit;
}


//******************************** LOCATE VALUE IN ARRAY *********************


int locate (double xx[], int n, int is, int ie, double xv)
{
    int jl, ju, jm, j;
//
// Initialize lower and upper methods:
//
    if (is<=0)
        is = 1;
    jl = is-1;
    ju = ie;
    if (xx[n]<=xv)
    {
        j = ie;
        return j;
    }
//
// If we are not done then compute a midpoint:
//
    while ((ju-jl)>1)
    {
        jm = int((ju+jl)/2);
//
// Replace the lower or upper limit with the midpoint:
//
        if((xx[ie]>xx[is])==(xv>xx[jm]))
            jl = jm;
        else
            ju = jm;
    }
//
// Return with the array index:
//
    j = jl;
    return j;
}


//******************** MAIN SEQUENTIAL INDICATOR SIMULATOR ****************


void sisim()
{
    double ntviol,atviol, TINY, xx, yy, zz, test, test2;
```

```
    int ic, is, ind, i, j, k, imult, nnz, nny, nnx, jz, jy, jx, iz, iy, ix, index, id, id2, index2, irepo, icut, in,
infoct;
    int nsec, nxysim, rt;
    double zval, cdfval, ntot, atot, btot;
    bool testind;
//
// Set up the rotation/anisotropy matrices that are needed for the
// variogram and search:
//
    ofstream outfile(outfl, ios::trunc);
    cout<<"Setting up rotation matrices for variogram and search"<<endl;
    for (ic=1;ic<=ncut;ic++)
    {
        for (is=1;is<=nst[ic];is++)
        {
            ind = is + (ic - 1)*MAXNST;
            setrot(ang1[ind], ang2[ind], ang3[ind], anis1[ind], anis2[ind], ind);
        }
    }
    isrot = MAXNST*MAXCUT + 1;
    setrot(sang1,sang2,sang3,sanis1,sanis2,isrot);
//
// Set up for super block searching:
//
    for (i=1;i<=nd;i++)
        actloc[i] = double(i);
    if (sstrat==0)
    {
        cout<<"Setting up super block search strategy"<<endl;
        nsec = 0;
        setsupr(nsec);
        picksup(isrot);
    }
//
//   Set up the covariance table and the spiral search:
//
    ctable();
//
//   Work out a random path for this realization:
//
    for (ind=1;ind<=nxyz;ind++)
    {
        sim[ind] = acorni();
        order[ind] = ind;
    }
//
//   The multiple grid search works with multiples of 4 (is
// arbitrary):
//
    if (mults==1)
    {
```

```
        for (imult=1;imult<=nmult;imult++)
        {
            nnz = __max(1,(nz/(imult*4)));
            nny = __max(1,(ny/(imult*4)));
            nnx = __max(1,(nx/(imult*4)));
            jz = 1;
            jy = 1;
            jx = 1;
            for (iz=1;iz<=nnz;iz++)
            {
                if (nnz>1)
                    jz = iz*imult*4;
                for (iy=1;iy<=nny;iy++)
                {
                    if (nny>1)
                        jy = iy*imult*4;
                    for (ix=1;ix<=nnx;ix++)
                    {
                        if(nnx>1)
                            jx = ix*imult*4;
                        index = jx + (jy-1)*nx + (jz-1)*nxy;
                        sim[index] -= imult;
                    }
                }
            }
        }
    }
    sortem(1,nxyz,sim,1,order,c,d,e,f,g,h);
//
//  Initialize the simulation:
//
    for (i=1;i<=nxyz;i++)
    {
        sim[i] = UNEST;
        tmp[i] = 0.0;
    }
//
//  INITIALIZE VECTOR OF HARD DATA FOR UPDATES
//
    for (i=1;i<=nxyz;i++)
        HD[i] = -9999.0;
//
//  Assign the data to the closest grid node:
//
    TINY = 0.0001;
    for (id=1;id<=nd;id++)
    {
        getindx(nx,xmn,xsiz,x[id],ix,testind);
        getindx(ny,ymn,ysiz,y[id],iy,testind);
        getindx(nz,zmn,zsiz,z[id],iz,testind);
        ind = ix + (iy-1)*nx + (iz-1)*nxy;
```

```
            xx = xmn + double(ix-1)*xsiz;
            yy = ymn + double(iy-1)*ysiz;
            zz = zmn + double(iz-1)*zsiz;
            test = fabs(xx-x[id]) + fabs(yy-y[id]) + fabs(zz-z[id]);
//
//  Assign this data to the node (unless there is a closer data):
//
            atnode[id] = false;
            if (sstrat==1)
                atnode[id] = true;
            if ((sstrat==0)&&(test<=TINY))
                atnode[id] = true;
            if (atnode[id])
            {
                if (sim[ind]>=0.0)
                {
                    id2 = int(sim[ind]+0.5);
                    index = int(actloc[id]+0.5);
                    index2 = int(actloc[id2]+0.5);
                    if (((index<=nhd)&&(index2<=nhd))||((index>nhd)&&(index2>nhd)))
                    {
                        test2 = fabs(xx-x[id2]) + fabs(yy-y[id2]) + fabs(zz-z[id2]);
                        if (test<=test2)
                        {
                            sim[ind] = double(id);
                            HD[ind] = vr[index][MXCUT];
                        }
                    }
                    else
                    {
                        if ((index<=nhd)&&(index2>nhd))
                        {
                            sim[ind] = double(id);
                            HD[ind] = vr[index][MXCUT];
                        }
                    }
                }
                else
                {
                    sim[ind] = double(id);
                    index = int(actloc[id]+0.5);
                    HD[ind] = vr[index][MXCUT];
                }
            }
        }
//
//  Now, enter the hard data values into the "sim" array and keep the
//  data number in the "tmp" array (to be reset when a hard value
//  is assigned to that node):
//
        for (i=1;i<=nxyz;i++)
```

```
    {
        id = int(sim[i]+0.5);
        if (id>0)
        {
            ind = int(actloc[id]+0.5);
            if (ind<=nhd)
                sim[i] = vr[ind][MXCUT];
            else
            {
                tmp[i] = sim[i];
                sim[i] = UNEST;
            }
        }
    }
//
//   Accumulate the number and magnitude of order relations violations:
//
    nclose = 0;
    irepo = __max(1,__min((nxyz/10),10000));
    ntviol = 0.0;
    atviol = 0.0;
    for (icut=1;icut<=ncut;icut++)
    {
        nviol[icut] = 0;
        aviol[icut] = 0.0;
        xviol[icut] = -1.0;
    }
    if (InRealType==1)
    {
//
// Load Initial Realization from Input file
//
        ifstream finit(InitFile, ios::nocreate);
        if (!finit)
        {
            cout<<"There is no Initial Realization file "<<InitFile<<endl;
            exit(1);
        }
        for (in=1;in<=nxyz;in++)
        {
            finit>>sim[in];
        }
        finit.close();
    }
    else
    {
//
//  Calculate the initial realization with Sisim
//   MAIN LOOP OVER ALL THE NODES:
//
        for (in=1;in<=nxyz;in++)
```

```
        {
            if ((int(in/irepo)*irepo)==in)
                cout<<"Currently on node "<<in<<endl;
            index = int(order[in]+0.5);
//
//  Do we really need to simulate this grid node location?
//
            if (sim[index]==UNEST)
            {
                if ((imbsim==0)&&(tmp[index]!=0.0))
                {
                    id = int(tmp[index]+0.5);
                    ind = int(actloc[id]+0.5);
                    sim[index] = vr[ind][MXCUT];
                }
                else
                {
//
//  Location of the node we are currently working on:
//
                    iz = int((index-1)/nxy) + 1;
                    iy = int((index-(iz-1)*nxy-1)/nx) + 1;
                    ix = index - (iz-1)*nxy - (iy-1)*nx;
                    xx = xmn + double(ix-1)*xsiz;
                    yy = ymn + double(iy-1)*ysiz;
                    zz = zmn + double(iz-1)*zsiz;
//
//  Now, we'll simulate the point ix,iy,iz.  First, get the close data
//  and make sure that there are enough to actually simulate a value,
//  we'll only keep the closest "ndmax" data, and look for previously
//  simulated grid nodes:
//
                    if (sstrat==0)
                    {
                        srchsuprsisim(xx,yy,zz,isrot,infoct);
                        if (nclose>ndmax)
                            nclose = ndmax;
                    }
                    ncnode = srchnd(ix,iy,iz);
//
//  What cdf value are we looking for?
//
                    zval = UNEST;
                    cdfval = draw[index]; //acorni();
//
//  Use the global distribution?
//
                    if ((nclose+ncnode)<=0)
                    {
                        beyond(cdf,zval,cdfval,ierr);
                    }
```

```
                    else
                    {
//
//   Estimate the local distribution by indicator kriging:
//
                        for (ic=1;ic<=ncut;ic++)
                            ccdf[ic] = krige(ix,iy,iz,xx,yy,zz,ic,cdf[ic]);
//
//   Correct order relations:
//
                        ordrel();
//
//Draw from the local distribution:
//
                        beyond(ccdfo,zval,cdfval,ierr);
                    }
                    sim[index] = zval;
                }
            }
//
//   END MAIN LOOP OVER NODES:
//
            tmp[index] = 0.0;
        }
    }
//
//   Write this simulation to the output file:
//
    nxysim = 0;
    for (ic=1;ic<=ncut;ic++)
        ccdf[ic] = 0.0;
    if (VarType==0)
        outfile<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<" /"<<endl<<endl<<"PERMX";
    ofstream outINC1("DiscPerm.INC", ios::trunc);
    ofstream outINC2("DiscPoro.INC", ios::trunc);
    if (VarType==1) {
        outfile<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<"
/"<<endl<<endl<<"SATNUM";
        outINC1<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<"
/"<<endl<<endl<<"PERMX";
        outINC2<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<" /"<<endl<<endl<<"PORO";
    }
    for (k=nz;k>=1;k--)
    {
        for (j=ny;j>=1;j--)
        {
            for (i=1;i<=nx;i++)
            {
                ind = i + (j-1)*nx + (k-1)*nxy;
                outfile<<endl<<sim[ind];
                if (VarType==1) {
```

```
                    rt = int(sim[ind]);
                    outINC1<<endl<<PermD[rt];
                    outINC2<<endl<<PoroD[rt];
                }
            }
        }
    }
    for (ind=1;ind<=nxyz;ind++)
    {
//
//  Calculate the cdf of the simulated values (for error checking):
//
        if (sim[ind]>UNEST)
        {
            nxysim++;
            clss[ind] = 0;
            for (ic=1;ic<=ncut;ic++)
            {
                if (ivtype==0)
                {
                    if (sim[ind]==thres[ic])
                    {
                        ccdf[ic] += 1.0;
                        clss[ind] = ic;
                    }
                }
                else
                {
                    if (sim[ind]<=thres[ic])
                    {
                        ccdf[ic] += 1.0;
                        if (clss[ind]==0)
                            clss[ind] = ic;
                    }
                }
            }
            if (clss[ind]==0)
                clss[ind] = 99;
        }
    }
    outfile<<" /"<<endl<<endl<<"ENDINC";
    if (VarType==1) {
        outINC1<<" /"<<endl<<endl<<"ENDINC";
        outINC2<<" /"<<endl<<endl<<"ENDINC";
    }
    outINC1.close();
    outINC2.close();
//
//  Report on the reproduction of the cdf and the number and magnitude
//  of order relations violations:
//
```

```
    for (icut=1;icut<=ncut;icut++)
    {
        ccdf[icut] = ccdf[icut]/double(__max(nxysim,1));
        cout<<"threshold: "<<icut<<", input cdf = "<<cdf[icut]<<" realization cdf =
"<<ccdf[icut]<<endl;
    }
    cout<<endl<<"Summary of order relations:"<<endl<<endl;
    ntot = 0;
    atot = 0.0;
    for (icut=1;icut<=ncut;icut++)
    {
        ntot += nviol[icut];
        atot += aviol[icut];
        aviol[icut] = aviol[icut]/double(__max(1,nviol[icut]));
        cout<<"threshold "<<icut<<", number = "<<nviol[icut]<<", average = "<<aviol[icut]<<",
maximum = "<<xviol[icut]<<endl;
    }
    atot = atot/double(__max(1,ntot));
    btot = (ntot/double(ncut*nxysim))*100.0;
    cout<<"Total of "<<btot<<"% with average of "<<atot<<endl;
//
//  END MAIN LOOP OVER SIMULATIONS:
//
    outfile.close();
//
//  Return to the main program:
//
    return;
}


//****************** SET ROTATION MATRIX - ANISOTROPY ************************


void setrot(double ag1, double ag2, double ag3, double ais1, double ais2, int ind)
{
    double DEG2RAD=3.141592654/180.0, EPSLON = 1.0e-20;
    double afac1, afac2, sina, sinb, sint, cosa, cosb, cost, alpha, beta, theta;
//
// Converts the input angles to three angles which make more
//  mathematical sense:
//
//        alpha   angle between the major axis of anisotropy and the
//                E-W axis. Note: Counter clockwise is positive.
//        beta    angle between major axis and the horizontal plane.
//                (The dip of the ellipsoid measured positive down)
//        theta   Angle of rotation of minor axis about the major axis
//                of the ellipsoid.
//
    if ((ag1>=0.0) && (ag1<270.0))
        alpha = (90.0-ag1)*DEG2RAD;
```

```
    else
        alpha = (450.0-ag1)*DEG2RAD;
    beta = -1.0*ag2*DEG2RAD;
    theta = ag3*DEG2RAD;
//
// Get the required sines and cosines:
//
    sina = sin(alpha);
    sinb = sin(beta);
    sint = sin(theta);
    cosa = cos(alpha);
    cosb = cos(beta);
    cost = cos(theta);
//
// Construct the rotation matrix in the required memory:
//
    afac1 = 1.0/__max(ais1,EPSLON);
    afac2 = 1.0/__max(ais2,EPSLON);
    rotmat[ind][1][1] = cosb*cosa;
    rotmat[ind][1][2] = cosb*sina;
    rotmat[ind][1][3] = -sinb;
    rotmat[ind][2][1] = afac1*(-cost*sina + sint*sinb*cosa);
    rotmat[ind][2][2] = afac1*(cost*cosa + sint*sinb*sina);
    rotmat[ind][2][3] = afac1*(sint*cosb);
    rotmat[ind][3][1] = afac2*(sint*sina + cost*sinb*cosa);
    rotmat[ind][3][2] = afac2*(-sint*cosa + cost*sinb*sina);
    rotmat[ind][3][3] = afac2*(cost*cosb);
//
// Return to calling program:
//
    return;
}


//********************* SET SUPER BLOCK SEARCH ***************************


void setsupr(int nsec)
{
    bool inflag;
    int i, ix, iy, iz, ii, nsort;
//
//   Establish the number and size of the super blocks:
//
    nxsup = __min(nx,MAXSBX);
    nysup = __min(ny,MAXSBY);
    nzsup = __min(nz,MAXSBZ);
    xsizsup = double(nx)*xsiz/double(nxsup);
    ysizsup = double(ny)*ysiz/double(nysup);
    zsizsup = double(nz)*zsiz/double(nzsup);
    xmnsup = (xmn-0.5*xsiz)+0.5*xsizsup;
```

```
    ymnsup = (ymn-0.5*ysiz)+0.5*ysizsup;
    zmnsup = (zmn-0.5*zsiz)+0.5*zsizsup;
//
//  Initialize the extra super block array to zeros:
//
    for (i=1;i<=(nxsup*nysup*nzsup);i++)
        nisb[i] = 0;
//
//  Loop over all the data assigning the data to a super block and
//  accumulating how many data are in each super block:
//
    for (i=1;i<=nd;i++)
    {
        getindx(nxsup,xmnsup,xsizsup,x[i],ix,inflag);
        getindx(nysup,ymnsup,ysizsup,y[i],iy,inflag);
        getindx(nzsup,zmnsup,zsizsup,z[i],iz,inflag);
        ii = ix + (iy-1)*nxsup + (iz-1)*nxsup*nysup;
        tmp[i] = ii;
        nisb[ii]++;
    }
//
//  Sort the data by ascending super block number:
//
    nsort = 4 + nsec;
    sortem(1,nd,tmp,nsort,x,y,z,vr,sec1,sec2,sec3);
//
//  Set up array nisb with the starting address of the block data:
//
    for (i=1;i<=(nxsup*nysup*nzsup-1);i++)
        nisb[i+1] += nisb[i];
    return;
}


//***************** GET GRID INDEX ******************************************


void getindx (int n, double min, double siz, double loc, int &index, bool&inflag)
{
    index = int((loc-min)/siz + 1.5);
    if (index<1)
    {
        index  = 1;
        inflag = false;
    }
    else
    {
        if (index>n)
        {
            index  = n;
            inflag = false;
```

```
        }
        else
         inflag = true;
    }
    return;
}


//***************** PICK SUPER BLOCK ************************************


void picksup(int irot)
{
    double hsqd, shortest, xo, yo, zo, xdis, ydis, zdis;
    int i, j, k, i1, j1, k1, i2, j2, k2;
//
//   MAIN Loop over all possible super blocks:
//
    nsbtosr = 0;
    for (i=-(nxsup-1);i<=(nxsup-1);i++)
    {
        for (j=-(nysup-1);j<=(nysup-1);j++)
        {
            for (k=-(nzsup-1);k<=(nzsup-1);k++)
            {
                xo = double(i)*xsizsup;
                yo = double(j)*ysizsup;
                zo = double(k)*zsizsup;
//
//   Find the closest distance between the corners of the super blocks:
//
                shortest = 1.0e21;
                for (i1=-1;i1<=1;i1++)
                {
                    for (j1=-1;j1<=1;j1++)
                    {
                        for (k1=-1;k1<=1;k1++)
                        {
                            for (i2=-1;i2<=1;i2++)
                            {
                                for (j2=-1;j2<=1;j2++)
                                {
                                    for (k2=-1;k2<=1;k2++)
                                    {
                                        if ((i1!=0)&&(j1!=0)&&(k1!=0)&&(i2!=0)&&(j2!=0)&&(k2!=0))
                                        {
                                            xdis = double(i1-i2)*0.5*xsizsup + xo;
                                            ydis = double(j1-j2)*0.5*ysizsup + yo;
                                            zdis = double(k1-k2)*0.5*zsizsup + zo;
                                            hsqd = sqdist(0.0,0.0,0.0,xdis,ydis,zdis,irot);
                                            if (hsqd<shortest)
```

```
                                        shortest = hsqd;
                                    }
                                }
                            }
                        }
                    }
                }
//
//   Keep this super block if it is close enoutgh:
//
                if (shortest<=radsqd)
                {
                    nsbtosr++;
                    ixsbtosr[nsbtosr] = i;
                    iysbtosr[nsbtosr] = j;
                    izsbtosr[nsbtosr] = k;
                }
            }
        }
    }
    return;
}


//************************** SQUARE DISTANCE *********************************


double sqdist(double x1, double y1, double z1, double x2, double y2, double z2, int ind)
{
    double sd, cont, dx, dy, dz;
    int i;
    dx = x1-x2;
    dy = y1-y2;
    dz = z1-z2;
    sd=0.0;
    for (i=1;i<=3;i++)
    {
        cont = rotmat[ind][i][1]*dx + rotmat[ind][i][2]*dy + rotmat[ind][i][3]*dz;
        sd+=cont*cont;
    }
    return sd;
}


//*************** CREATE COVARIANCE LOOK UP TABLE ***************************


void ctable()
{
    double TINY=1.0e-10;
```

```
    double hsqd, cbb, cmx, xx, yy, zz, cov;
    int ilooku, icut, irot, i, ic, jc, kc, j, k, il, loc, iz, iy, ix;
//
//  Size of the look-up table:
//
    nctx = __min(((MAXCTX-1)/2),(nx-1));
    ncty = __min(((MAXCTY-1)/2),(ny-1));
    nctz = __min(((MAXCTZ-1)/2),(nz-1));
//
//  Initialize the covariance subroutine and cbb at the same time:
//
    cbb = cova3(0.,0.,0.,0.,0.,0.,1,1,cmx);
//
//  Now, set up the table and keep track of the node offsets that are
//  within the search radius:
//
    ilooku = __max((ncut/2),1);
    nlooku = 0;
    for (icut=1;icut<=ncut;icut++)
    {
        irot = 1 + (icut-1)*MAXNST;
        for (i=-nctx;i<=nctx;i++)
        {
            xx = i*xsiz;
            ic = nctx + 1 + i;
            for (j=-ncty;j<=ncty;j++)
            {
                yy = j*ysiz;
                jc = ncty + 1 + j;
                for (k=-nctz;k<=nctz;k++)
                {
                    zz = k*zsiz;
                    kc = nctz + 1 + k;
                    cov = cova3(0.,0.,0.,xx,yy,zz,icut,irot,cmx);
                    covtab[ic][jc][kc][icut] = cov;
                    if (icut==ilooku)
                    {
                        hsqd = sqdist(0.0,0.0,0.0,xx,yy,zz,isrot);
                        if (hsqd<=radsqd)
                        {
                            nlooku++;
//
//  We subtract the covariance from a large value so that the ascending
//  sort subroutine will accomplish the sort we want.  Furthermore, a
//  fraction of the distance is also taken off so that we search by
//  anisotropic distance once we are beyond the range:
//
                            tmp[nlooku] = -(covtab[ic][jc][kc][icut]-TINY*hsqd);
                            order[nlooku] = double((kc-1)*MAXCXY+(jc-1)*MAXCTX+ic);
                        }
                    }
```

```
                }
            }
        }
    }
//
//  Finished setting up the look-up table, now order the nodes such
//  that the closest ones, according to variogram distance, are searched
//  first. Note: the "loc" array is used because I didn't want to make
//  special allowance for 2 byte integers in the sorting subroutine:
//
    sortem(1,nlooku,tmp,1,order,c,d,e,f,g,h);
    for (il=1;il<=nlooku;il++)
    {
        loc = int(order[il]);
        iz = int((loc-1)/MAXCXY) + 1;
        iy = int((loc-(iz-1)*MAXCXY-1)/MAXCTX) + 1;
        ix = loc-(iz-1)*MAXCXY - (iy-1)*MAXCTX;
        iznode[il] = iz;
        iynode[il] = iy;
        ixnode[il] = ix;
    }
    if (nodmax>MAXNOD)
    {
        nodmax = MAXNOD;
    }
//
//  Debugging output if requested:
//
    if (idbg<=2)
        return;
    if (idbg<4)
        return;
    for (i=1;i<=nlooku;i++)
    {
        xx = (ixnode[i] - nctx - 1)*xsiz;
        yy = (iynode[i] - ncty - 1)*ysiz;
        zz = (iznode[i] - nctz - 1)*zsiz;
    }
//
//  All finished:
//
    return;
}


//********************* ESTIMATE COVARIANCE 3D ****************************


double cova3(double x1, double y1, double z1, double x2, double y2, double z2, int ivarg, int irot,
double& cmx)
{
```

```
    double PI=3.14159265, PMX=999., EPSLON=1.e-10;
    double cova, hsqd, h, hr;
    int istart, is, ist, ir;
//
//  Calculate the maximum covariance value (used for zero distances and
//  for power model covariance):
//
    istart = 1 + (ivarg-1)*MAXNST;
    cmx = c0[ivarg];
    for (is=1;is<=nst[ivarg];is++)
    {
        ist = istart + is - 1;
        if (it[ist]==4)
            cmx += PMX;
        else
            cmx += cc[ist];
    }
//
//  Check for "zero" distance, return with cmx if so:
//
    hsqd = sqdist(x1,y1,z1,x2,y2,z2,irot);
    if (hsqd<EPSLON)
        cova = cmx;
//
// Loop over all the structures:
//
    else
    {
        cova = 0.0;
        for (is=1;is<=nst[ivarg];is++)
        {
            ist = istart + is - 1;
//
//  Compute the appropriate distance:
//
            if (ist!=1)
            {
                ir = __min((irot+is-1),MAXROT);
                hsqd = sqdist(x1,y1,z1,x2,y2,z2,ir);
            }
            h = sqrt(hsqd);
//
//  Spherical Variogram Model?
//
            if (it[ist]==1)
            {
                hr = h/aa[ist];
                if (hr<1)
                    cova += cc[ist]*(1.0-hr*(1.5-0.5*hr*hr));
            }
//
```

```
//   Exponential Variogram Model?
//
            else
            {
                if (it[ist]==2)
                    cova += cc[ist]*exp(-3.0*h/aa[ist]);
//
//   Gaussian Variogram Model?
//
                else
                {
                    if (it[ist]==3)
                        cova += cc[ist]*exp(-3.0*(h/aa[ist])*(h/aa[ist]));
//
//   Power Variogram Model?
//
                    else
                    {
                        if (it[ist]==4)
                            cova += cmx-cc[ist]*pow(h,aa[ist]);
//
//   Hole Effect Model?
//
                        else
                        {
                            if (it[ist]==5)
                                cova += cc[ist]*cos(h/aa[ist]*2.0*PI);
                        }
                    }
                }
            }
        }
    }
    return cova;
}


//****************** SEARCH SUPER BLOCK ***********************************


void srchsuprsisim(double xloc, double yloc,double zloc, int irot, int &infoct)
{
    double hsqd, dx, dy, dz, hh;
    int inoct[9], ix, iy, iz, isup, ixsup, iysup, izsup, ii, nums, i, nclose2, nsoft, ind, nt, na, j, iq;
    bool inflag;
//
//   Determine the super block location of point being estimated:
//
    getindx(nxsup,xmnsup,xsizsup,xloc,ix,inflag);
    getindx(nysup,ymnsup,ysizsup,yloc,iy,inflag);
    getindx(nzsup,zmnsup,zsizsup,zloc,iz,inflag);
```

```
//
//   Loop over all the possible Super Blocks:
//
    nclose = 0;
    for (isup=1;isup<=nsbtosr;isup++)
    {
//
//   Is this super block within the grid system:
//
        ixsup = ix + ixsbtosr[isup];
        iysup = iy + iysbtosr[isup];
        izsup = iz + izsbtosr[isup];
        if
((ixsup>0)&&(ixsup<=nxsup)&&(iysup>0)&&(iysup<=nysup)&&(izsup>0)&&(izsup<=nzsup))
        {
//
//   Figure out how many samples in this super block:
//
            ii = ixsup + (iysup-1)*nxsup + (izsup-1)*nxsup*nysup;
            if (ii==1)
            {
                nums = nisb[ii];
                i = 0;
            }
            else
            {
                nums = nisb[ii] - nisb[ii-1];
                i = nisb[ii-1];
            }
//
//   Loop over all the data in this super block:
//
            for (ii=1;ii<=nums;ii++)
            {
                i++;
//
//   Check squared distance:
//
                hsqd = sqdist(xloc,yloc,zloc,x[i],y[i],z[i],irot);
                if (hsqd<=radsqd)
                {
//
//Accept this sample:
//
                    nclose++;
                    close[nclose] = double(i);
                    tmpdat[nclose] = hsqd;
                }
            }
        }
    }
```

```
//
//   Sort the nearby samples by distance to point being estimated:
//
     sortem(1,nclose,tmpdat,1,close,c,d,e,f,g,h);
//
//   Retain less than maxsec soft data
//
     nclose2 = nclose;
     nclose = 0;
     nsoft = 0;
     for (i=1;i<=nclose2;i++)
     {
          ind = int(close[i]);
          if (int(actloc[ind])>nhd)
               nsoft++;
          if ((int(actloc[ind])<=nhd)||(nsoft<=maxsec))
          {
               nclose++;
               close[nclose] = double(ind);
          }
     }
//
//   If we aren't doing an octant search then just return:
//
     if (noct<=0)
          return;
//
//   PARTITION THE DATA INTO OCTANTS:
//
     for (i=1;i<=8;i++)
          inoct[i] = 0;
//
//   Now pick up the closest samples in each octant:
//
     nt = 8*noct;
     na = 0;
     j = 0;
     while ((j<nclose)&&(na<nt))
     {
          j++;
          i = int(close[j]);
          hh = tmpdat[j];
          dx = x[i] - xloc;
          dy = y[i] - yloc;
          dz = z[i] - zloc;
          if (dz>=0.)
          {
               iq = 4;
               if ((dx<=0.0)&&(dy>0.0))
                    iq = 1;
               if ((dx>0.0)&&(dy>=0.0))
```

```
                iq = 2;
            if ((dx<0.0)&&(dy<=0.0))
                iq = 3;
        }
        else
        {
            iq = 8;
            if ((dx<=0.0)&&(dy>0.0))
                iq = 5;
            if ((dx>0.0)&&(dy>=0.0))
                iq = 6;
            if ((dx<0.0)&&(dy<=0.0))
                iq = 7;
        }
        inoct[iq]++;
//
//  Keep this sample if the maximum has not been exceeded:
//
        if (inoct[iq]<=noct)
        {
            na++;
            close[na] = i;
            tmpdat[na] = hh;
        }
    }
//
//  End of data selection. Compute number of informed octants and return:
//
    nclose = na;
    infoct = 0;
    for (i=1;i<=8;i++)
    {
        if (inoct[i]>0)
            infoct++;
    }
//
//  Finished:
//
    return;
}


//****************** SEARCH COND DATA LOCATION *********************


int srchnd(int ix, int iy, int iz)
{
    int ncnd, ncsec, il, i, j, k, index;
//
//  Consider all the nearby nodes until enough have been found:
//
```

```
    ncnd = 0;
    ncsec = 0;
    for (il=1;il<=nlooku;il++)
    {
        if (ncnd==nodmax)
            return ncnd;
        i = ix + (int(ixnode[il])-nctx-1);
        if ((i>=1)&&(i<=nx))
        {
            j = iy + (int(iynode[il])-ncty-1);
            if ((j>=1)&&(j<=ny))
            {
                k = iz + (int(iznode[il])-nctz-1);
                if ((k>=1)&&(k<=nz))
                {
//
//  Check this potentially informed grid node:
//
                    index = (k-1)*nx*ny + (j-1)*nx + i;
                    if ((sim[index]>UNEST)||(tmp[index]>0.5))
                    {
                        if ((sim[index]<=UNEST)&&(tmp[index]>0.5))
                            ncsec++;
                        if (ncsec<=maxsec)
                        {
                            ncnd++;
                            icnode[ncnd] = il;
                            cnodex[ncnd] = xmn + double(i-1)*xsiz;
                            cnodey[ncnd] = ymn + double(j-1)*ysiz;
                            cnodez[ncnd] = zmn + double(k-1)*zsiz;
                            cnodev[ncnd] = sim[index];
                            cnodet[ncnd] = tmp[index];
                        }
                    }
                }

            }
        }
    }
//
//  Return to calling program:
//
    return ncnd;
}


//********************** PERFORM KRIGING ***********************************


double krige(int ix, int iy, int iz, double xx, double yy, double zz, int icut, double gmean)
{
```

```
    int aclose[MAXKR1+1], mclose, i, index, ind, na, neq, irot, in, j1, j, iii, ix1, iy1, iz1, ix2, iy2, iz2,
ii, jj, kk, ising;
    double x1, y1, z1, cmean, x2, y2, z2, cov, cmx, sumwt;
    bool krig, somesoft, bothsoft;
//
//  Size of the kriging system:  Some of the data values may be missing
//  which would correspond to a constraint interval.  Note that there
//  should not be any missing values if the median approximation is being
//  considered.  The variable ``krig" is used
//  to flag whether kriging is to be done or if the previous weights are
//  to be used.
//
    somesoft = false;
    krig = true;
    if ((mik==1)&&(icut>1))
        krig = false;
    if (krig)
    {
        mclose = 0;
        for (i=1;i<=nclose;i++)
        {
            index =  int(close[i]);
            ind = int(actloc[index]+0.5);
            if ((!atnode[index])&&(vr[ind][icut]>=0.0))
            {
                mclose++;
                aclose[mclose] = index;
            }
        }
        na = mclose + ncnode;
        neq = na + ktype;
    }
//
//  There are no data yet:
//
    irot = 1 + (icut-1)*MAXNST;
//
//  Set up kriging matrices:
//
    in = 0;
    j1 = 0;
    for (j=1;j<=na;j++)
    {
        softdat[j] = false;
//
//  Sort out the actual location of point "j"
//
        if (j<=mclose)
        {
            index = aclose[j];
            ind = int(actloc[index]+0.5);
```

```
            vra[j] = vr[ind][icut];
            x1 = x[index];
            y1 = y[index];
            z1 = z[index];
            if (ind>nhd)
                softdat[j] = true;
        }
        else
        {
//
//  It is a previously simulated node (keep index for table look-up):
//
            index = j - mclose;
            x1 = cnodex[index];
            y1 = cnodey[index];
            z1 = cnodez[index];
//
//  Is this node informed by a hard datum or a soft datum?
//
            if (cnodet[index]<=0.5)
            {
                if (ivtype==0)
                {
                    vra[j] = 0.0;
                    if (int(cnodev[index]+0.5)==int(thres[icut]+0.5))
                        vra[j] = 1.0;
                }
                else
                {
                    vra[j] = 1.0;
                    if (cnodev[index]>thres[icut])
                        vra[j] = 0.0;
                }
            }
            else
            {
                iii = int(cnodet[index]+0.5);
                ind = int(actloc[iii]+0.5);
                vra[j] = vr[ind][icut];
                softdat[j] = true;
            }
            ind = icnode[index];
            ix1 = ix + (int(ixnode[ind])-nctx-1);
            iy1 = iy + (int(iynode[ind])-ncty-1);
            iz1 = iz + (int(iznode[ind])-nctz-1);
        }
//
//  Only set up the matrix and the RHS if kriging:
//
        if (krig)
        {
```

```
            for (i=1;i<=j;i++)
            {
//
//  Sort out the actual location of point "i"
//
                if (i<=mclose)
                {
                    index = aclose[i];
                    x2 = x[index];
                    y2 = y[index];
                    z2 = z[index];
                }
                else
                {
//
//  It is a previously simulated node (keep index for table look-up):
//
                    index = i - mclose;
                    x2 = cnodex[index];
                    y2 = cnodey[index];
                    z2 = cnodez[index];
                    ind = icnode[index];
                    ix2 = ix + (int(ixnode[ind])-nctx-1);
                    iy2 = iy + (int(iynode[ind])-ncty-1);
                    iz2 = iz + (int(iznode[ind])-nctz-1);
                }
//
//  Now, get the covariance value:
//
                in++;
//
//  Decide whether or not to use the covariance look-up table:
//
                if ((j<=mclose)||(i<=mclose))
                {
                    cov = cova3(x1,y1,z1,x2,y2,z2,icut,irot,cmx);
                    a[in] = cov;
                }
                else
                {
//
//  Try to use the covariance look-up (if the distance is in range):
//
                    ii = nctx + 1 + (ix1 - ix2);
                    jj = ncty + 1 + (iy1 - iy2);
                    kk = nctz + 1 + (iz1 - iz2);
                    if ((ii<1)||(ii>MAXCTX)||(jj<1)||(jj>MAXCTY)||(kk<1)||(kk>MAXCTZ))
                    {
                        cov = cova3(x1,y1,z1,x2,y2,z2,icut,irot,cmx);
                        a[in] = cov;
                    }
```

```
                    else
                        a[in] = covtab[ii][jj][kk][icut];
                }
            }
//
//  Get the RHS value (possibly with covariance look-up table):
//
            if (j<=mclose)
            {
                cov = cova3(xx,yy,zz,x1,y1,z1,icut,irot,cmx);
                r[j] = cov;
            }
            else
            {
//
//  Try to use the covariance look-up (if the distance is in range):
//
                ii = nctx + 1 + (ix - ix1);
                jj = ncty + 1 + (iy - iy1);
                kk = nctz + 1 + (iz - iz1);
                if ((ii<1)||(ii>MAXCTX)||(jj<1)||(jj>MAXCTY)||(kk<1)||(kk>MAXCTZ))
                {
                    cov = cova3(xx,yy,zz,x1,y1,z1,icut,irot,cmx);
                    r[j] = cov;
                }
                else
                    r[j] = covtab[ii][jj][kk][icut];
            }
            rr[j] = r[j];
//
//  End ``if'' block (true if kriging)
//
        }
//
//  End loop over all of the nearby data
//
        if (softdat[j])
            somesoft = true;
    }
//
//  If we are doing Markov-Bayes are there are soft data we need to
//  correct some of the covariance values in the kriging matrix:
//
    if ((imbsim==1)&&(somesoft))
    {
        in = 0;
        for (j=1;j<=na;j++)
        {
            for (i=1;i<=j;i++)
            {
                in++;
```

```
                bothsoft = false;
                if ((softdat[j])&&(softdat[i]))
                    bothsoft = true;
//
//  Correct for soft-soft covariance or soft-hard covariance:
//
                if (bothsoft)
                {
                    a[in] = a[in]*beez[icut];
                    if (i!=j)
                        a[in] = a[in]*beez[icut];
                }
                else
                {
                    if ((softdat[j])||(softdat[i]))
                        a[in] = a[in]*beez[icut];
                }
            }
//
//  Correct the right hand side for soft-hard covariance:
//
            if (softdat[j])
            {
                r[j] = r[j]*beez[icut];
                rr[j] = r[j];
            }
        }
    }
//
//  Addition of OK constraint:
//
    if ((krig)&&(ktype==1))
    {
        for (i=1;i<=na;i++)
        {
            in++;
            a[in] = 1.0;
        }
        in++;
        a[in] = 0.0;
        r[neq] = 1.0;
        rr[neq] = 1.0;
    }
//
//  Solve the Kriging System:
//
    if (krig)
    {
        if ((neq==1)&&(ktype==0))
        {
            s[1] = r[1]/a[1];
```

```
            ising = 0;
        }
        else
            ksol(1,neq,1,ising);
    }
//
//  Write a warning if the matrix is singular:
//
    if (ising!=0)
    {
        cmean = 0.0;
        return cmean;
    }
//
//  Compute the estimate, the sum of weights, correct for SK, and return:
//
    cmean = 0.0;
    sumwt = 0.0;
    for (i=1;i<=na;i++)
    {
        cmean += s[i]*vra[i];
        sumwt += s[i];
    }
    if (ktype==0)
        cmean += (1.0-sumwt)*gmean;
    return cmean;
}


//***************** SOLVE SYSTEM OF EQUATIONS ***********************************


void ksol(int nright, int neq, int nsb,int& ising)
{
    double tol, ak, piv, ap;
    int nn, nm, m1, kk, k, km1, iv, nm1, ii, lp, i, ll, ij, j, llb, in, ll1, ijm;
//
//  If there is only one equation then set ising and return:
//
    if (neq<=1)
    {
        ising = -1;
        return;
    }
//
//  Initialize:
//
    tol = 0.1e-06;
    ising = 0;
    nn = neq*(neq+1)/2;
    nm = nsb*neq;
```

```
    m1 = neq - 1;
    kk = 0;
//
//  Start triangulation:
//
    for (k=1;k<=m1;k++)
    {
        kk += k;
        ak = a[kk];
        if (fabs(ak)<tol)
        {
            ising = k;
            return;
        }
        km1 = k - 1;
        for (iv=1;iv<=nright;iv++)
        {
            nm1 = nm*(iv-1);
            ii = kk + nn*(iv-1);
            piv = 1./a[ii];
            lp = 0;
            for (i=k;i<=m1;i++)
            {
                ll = ii;
                ii = ii + i;
                ap = a[ii]*piv;
                lp++;
                ij = ii - km1;
                for (j=i;j<=m1;j++)
                {
                    ij += j;
                    ll += j;
                    a[ij] -= ap*a[ll];
                }
                for (llb=k;llb<=nm;llb+=neq)
                {
                    in = llb + lp + nm1;
                    ll1 = llb + nm1;
                    r[in] -= ap*r[ll1];
                }
            }
        }
    }
//
//  Error checking - singular matrix:
//
    ijm = ij - nn*(nright-1);
    if (fabs(a[ijm])<tol)
    {
        ising = neq;
        return;
```

```
    }
//
//   Finished triangulation, start solving back:
//
    for (iv=1;iv<=nright;iv++)
    {
        nm1 = nm*(iv-1);
        ij = ijm + nn*(iv-1);
        piv = 1./a[ij];
        for (llb=neq;llb<=nm;llb+=neq)
        {
            ll1 = llb + nm1;
            s[ll1] = r[ll1]*piv;
        }
        i = neq;
        kk = ij;
        for (ii=1;ii<=m1;ii++)
        {
            kk -= i;
            piv = 1./a[kk];
            i--;
            for (llb=i;llb<=nm;llb+=neq)
            {
                ll1 = llb + nm1;
                in = ll1;
                ap = r[in];
                ij = kk;
                for (j=i;j<=m1;j++)
                {
                    ij += j;
                    in++;
                    ap -= a[ij]*s[in];
                }
                s[ll1] = ap*piv;
            }
        }
    }
//
//   Finished solving back, return:
//
    return;
}


//***************** GRADUAL DEFORMATION SISIM ********************************


void updatedsisim()
{
    double xx, yy ,zz, rdt, TINY, test, test2;
    int ic, ind, i, iz, iy, ix, index, irepo, icut, in, infoct, id, id2, index2;
```

```
    int nxysim, j, k, rt;
    double zval, cdfval, cum, tot;
    bool testind;
    int cl;
    double sum, apr, bpr, cpr, max;
    double pdf[MAXCUT+1], spdf[MAXCUT+1], dpdf[MAXCUT+1], scdf[MAXCUT+1],
DCDF[MAXCUT+1], ocdf[MAXCUT+1];
    for (i=1;i<=nd;i++)
        actloc[i] = double(i);
    nclose = 0;
    irepo = __max(1,__min((nxyz/10),10000));
    bool perturb = true;
    if ((Subdomains>0)&&(iReg>Subdomains)){
        perturb = false;
        cout<<"Final Realization"<<endl;
    }
//
//  Initialize the simulation:
//
    for (i=1;i<=nxyz;i++)
    {
        if (perturb)
        {
            if ((Subdomains==0)||((Subdomains>0)&&(RegIndex[i]==RegionIndex[iReg])))
            {
                sim[i] = UNEST;
                tmp[i] = 0.0;
            }
        }
        else {
            if (RegIndex[i]<0)
            {
                sim[i] = UNEST;
                tmp[i] = 0.0;
            }
        }
    }
    TINY = 0.0001;
    for (id=1;id<=nd;id++)
    {
        getindx(nx,xmn,xsiz,x[id],ix,testind);
        getindx(ny,ymn,ysiz,y[id],iy,testind);
        getindx(nz,zmn,zsiz,z[id],iz,testind);
        ind = ix + (iy-1)*nx + (iz-1)*nxy;
        xx = xmn + double(ix-1)*xsiz;
        yy = ymn + double(iy-1)*ysiz;
        zz = zmn + double(iz-1)*zsiz;
        test = fabs(xx-x[id]) + fabs(yy-y[id]) + fabs(zz-z[id]);
//
//  Assign this data to the node (unless there is a closer data):
//
```

```
        atnode[id] = false;
        if (sstrat==1)
            atnode[id] = true;
        if ((sstrat==0)&&(test<=TINY))
            atnode[id] = true;
        if (atnode[id])
        {
            if (sim[ind]>=0.0)
            {
                id2 = int(sim[ind]+0.5);
                index = int(actloc[id]+0.5);
                index2 = int(actloc[id2]+0.5);
                if (((index<=nhd)&&(index2<=nhd))||((index>nhd)&&(index2>nhd)))
                {
                    test2 = fabs(xx-x[id2]) + fabs(yy-y[id2]) + fabs(zz-z[id2]);
                    if (test<=test2)
                    {
                        sim[ind] = double(id);
                    }
                }
                else
                {
                    if ((index<=nhd)&&(index2>nhd))
                    {
                        sim[ind] = double(id);
                    }
                }
            }
            else
            {
                sim[ind] = double(id);
                index = int(actloc[id]+0.5);
            }
        }
    }
//
//  Now, enter the hard data values into the "sim" array and keep the
//  data number in the "tmp" array (to be reset when a hard value
//  is assigned to that node):
//
    for (i=1;i<=nxyz;i++)
    {
        id = int(sim[i]+0.5);
        if (id>0)
        {
            ind = int(actloc[id]+0.5);
            if (ind<=nhd)
                sim[i] = vr[ind][MXCUT];
            else
            {
                tmp[i] = sim[i];
```

```
                sim[i] = UNEST;
            }
        }
    }
//
//  MAIN LOOP OVER ALL THE NODES:
//
    for (in=1;in<=nxyz;in++)
    {
        if ((int(in/irepo)*irepo)==in)
            cout<<"Currently on node "<<in<<endl;
        if (perturb) {
            if (rd<=0.25)
                index = int(order[in]+0.5);
            else {
                if (rd<=0.5)
                    index = int(order2[in]+0.5);
                else {
                    if (rd<=0.75)
                        index = int(order3[in]+0.5);
                    else
                        index = int(order4[in]+0.5);
                }
            }
        }
        else
            index = int(order[in]+0.5);
        if (sim[index]==UNEST)
        {
//
//  Location of the node we are currently working on:
//
                iz = int((index-1)/nxy) + 1;
                iy = int((index-(iz-1)*nxy-1)/nx) + 1;
                ix = index - (iz-1)*nxy - (iy-1)*nx;
                xx = xmn + double(ix-1)*xsiz;
                yy = ymn + double(iy-1)*ysiz;
                zz = zmn + double(iz-1)*zsiz;
//
//  Now, we'll simulate the point ix,iy,iz.  First, get the close data
//  and make sure that there are enough to actually simulate a value,
//  we'll only keep the closest "ndmax" data, and look for previously
//  simulated grid nodes:
//
                if (sstrat==0)
                {
                    srchsuprsisim(xx,yy,zz,isrot,infoct);
                    if (nclose>ndmax)
                        nclose = ndmax;
                }
                ncnode = srchnd(ix,iy,iz);
```

```
//
//   What cdf value are we looking for?
//
                zval = UNEST;
                if (perturb) {
                    if (rd<=0.25)
                        cdfval = draw[index];
                    else {
                        if (rd<=0.5)
                            cdfval = draw2[index];
                        else {
                            if (rd<=0.75)
                                cdfval = draw3[index];
                            else
                                cdfval = draw4[index];
                        }
                    }
                }
                else
                    cdfval = draw[index];
//
//   Use the global distribution?
//
                if ((nclose+ncnode)<=0)
                {
                    if (perturb) {
                        if (ivtype==0) {
                            cum = 0.0;
                            tot = 0.0;
                            for (i=1;i<=ncut;i++)
                                tot += cdf[i];
                            for (i=1;i<=ncut;i++) {
                                cum += cdf[i];
                                scdf[i] = cum/tot;
                            }
                        }
                        else {
                            for (ic=1;ic<=ncut;ic++)
                                scdf[ic] = cdf[ic];
                        }
                    }
                    else
                        beyond(cdf,zval,cdfval,ierr);
                }
                else
                {
//
//   Estimate the local distribution by indicator kriging:
//
                    for (ic=1;ic<=ncut;ic++) {
                        ccdf[ic] = krige(ix,iy,iz,xx,yy,zz,ic,cdf[ic]);
```

```
                    }
//
//  Correct order relations:
//
                    ordrel();
                    if (perturb) {
//
//  Save as static cdf
//
                        if (ivtype==0) {
                            cum = 0.0;
                            tot = 0.0;
                            for (i=1;i<=ncut;i++)
                                tot += ccdfo[i];
                            for (i=1;i<=ncut;i++) {
                                cum += ccdfo[i];
                                scdf[i] = cum/tot;
                            }
                        }
                        else {
                            for (ic=1;ic<=ncut;ic++)
                                scdf[ic] = ccdfo[ic];
                        }
                    }
                    else
                        beyond(ccdfo,zval,cdfval,ierr);
                }
                if (perturb) {
//
//  Determine the PDF from Static Information
//
                    for (ic=ncut;ic>=2;ic--)
                        spdf[ic] = scdf[ic] - scdf[ic-1];
                    spdf[1] = scdf[1];
//
//  Determine pdf of prior cdf
//
                    if (ivtype==0) {
                        tot = 0.0;
                        for (i=1;i<=ncut;i++)
                            tot += cdf[i];
                        for (ic=1;ic<=ncut;ic++)
                            pdf[ic] = cdf[ic]/tot;
                    }
                    else {
                        for (ic=ncut;ic>=2;ic--)
                            pdf[ic] = cdf[ic] - cdf[ic-1];
                        pdf[1] = cdf[1];
                    }
//
//  Transform for rd
```

```
//
                    if (rd<=0.25)
                        rdt = 1 - 4*rd;
                    else
                    {
                        if (rd>=0.75)
                            rdt = 4*rd - 3;
                        else
                        {
                            if (rd<=0.5)
                                rdt = 4*rd - 1;
                            else
                                rdt = 3 - 4*rd;
                        }
                    }
                    currentRDT = rdt;
//
//  Consider the probability for values higher than the last threshold.
//
                    sum = 0.0;
                    for (ic=1;ic<=ncut;ic++)
                        sum += pdf[ic];
                    if (sum<1.0)
                        max = (1 - sum)*rdt;
                    else
                        max = 0.0;
//
//  Find the Conditional Distribution to Production Data: pdf
//
                    cl = clss[index];
                    sum = 0.0;
                    for (ic=1;ic<=ncut;ic++)
                    {
                        if (ic!=cl)
                        {
                            dpdf[ic] = rdt*pdf[ic];
                            sum += dpdf[ic];
                        }
                    }
                    if (cl!=99)
//
// cl = 99 when value is higher than highest threshold
//
                        dpdf[cl] = 1.0 - sum - max;
//
//  Obtain the CDF from the pdf
//
                    DCDF[1] = dpdf[1];
                    for (ic=2;ic<=ncut;ic++)
                    {
                        DCDF[ic] = dpdf[ic] + DCDF[ic-1];
```

```
                    }
                    if (ivtype==0) {
                        ocdf[1] = cdf[1];
                        for (i=2;i<=ncut;i++)
                            ocdf[i] = cdf[i] + ocdf[i-1];
                    }
//
//  Combine the Dinamic and Static Conditional Distribution - Permanence of Ratio
//
                    if (rdt==1) {
                        for (ic=1;ic<=ncut;ic++)
                            ccdf[ic] = scdf[ic];
                    }
                    else {
                        for (ic=1;ic<=ncut;ic++)
                        {
                            if (ivtype==0)
                                apr = (1 - ocdf[ic])/ocdf[ic];
                            else
                                apr = (1 - cdf[ic])/cdf[ic];
                            bpr = (1 - scdf[ic])/scdf[ic];
                            cpr = (1 - DCDF[ic])/DCDF[ic];
                            ccdf[ic] = apr/(apr+bpr*cpr);
                        }
                        if (ivtype==0)
                            ccdf[ncut] = 1;
                    }
//
//  Correct order relationships
//
                    if (ivtype==0) {
                        ivtype = 1;
                        ordrel();
                        ivtype = 0;
                    }
                    else
                        ordrel();
                    if (ivtype==0) {
                        for (ic=ncut;ic>=2;ic--)
                            ccdfo[ic] = ccdfo[ic] - ccdfo[ic-1];
                        ccdfo[1] = ccdfo[1];
                    }
//
//Draw from the local distribution:
//
                        beyond(ccdfo,zval,cdfval,ierr);
                }
                sim[index] = zval;
        }
//
//  END MAIN LOOP OVER NODES:
```

```
//
        tmp[index] = 0.0;
    }
//
//  Write this simulation to the output file:
//
    if (rdt==1)
        cout<<"rdt="<<rdt<<"; rd="<<rd<<endl;
    nxysim = 0;
    for (ic=1;ic<=ncut;ic++)
        ccdf[ic] = 0.0;
    for (ind=1;ind<=nxyz;ind++)
    {
//
//  Calculate the cdf of the simulated values (for error checking):
//
        if (sim[ind]>UNEST)
        {
            nxysim++;
            for (ic=1;ic<=ncut;ic++)
            {
                if (ivtype==0)
                {
                    if (sim[ind]==thres[ic])
                        ccdf[ic] += 1.0;
                }
                else
                {
                    if (sim[ind]<=thres[ic])
                        ccdf[ic] += 1.0;
                }
            }
        }
    }
//
//  Report on the reproduction of the cdf
//
    for (icut=1;icut<=ncut;icut++)
    {
        ccdf[icut] = ccdf[icut]/double(__max(nxysim,1));
        cout<<"threshold: "<<icut<<", input cdf = "<<cdf[icut]<<" realization cdf =
"<<ccdf[icut]<<endl;
    }
    ofstream outfile2(outfl, ios::trunc);
    ofstream outINC3("DiscPerm.INC", ios::trunc);
    ofstream outINC4("DiscPoro.INC", ios::trunc);
    if (VarType==0)
        outfile2<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<"
/"<<endl<<endl<<"PERMX";
    if (VarType==1) {
```

```
        outfile2<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<"
/"<<endl<<endl<<"SATNUM";
        outINC3<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<"
/"<<endl<<endl<<"PERMX";
        outINC4<<"BOX"<<endl<<"1 "<<nx<<" 1 "<<ny<<" 1 "<<nz<<" /"<<endl<<endl<<"PORO";
    }
    for (k=nz;k>=1;k--)
    {
        for (j=ny;j>=1;j--)
        {
            for (i=1;i<=nx;i++)
            {
                ind = i + (j-1)*nx + (k-1)*nxy;
                outfile2<<endl<<sim[ind];
                if (VarType==1) {
                    rt = int(sim[ind]);
                    outINC3<<endl<<PermD[rt];
                    outINC4<<endl<<PoroD[rt];
                }
            }
        }
    }
    outfile2<<" /"<<endl<<endl<<"ENDINC";
    if (VarType==1) {
        outINC3<<" /"<<endl<<endl<<"ENDINC";
        outINC4<<" /"<<endl<<endl<<"ENDINC";
    }
    outfile2.close();
    outINC3.close();
    outINC4.close();
    return;
}


//***************** READ FILE WITH PRODUCTION HISTORY *********************


void readProdHist()
{
    int i, j;
    double sum;
    char tempch[80], tempc[10];
//
//  Read Input Parameters:
//
    ifstream prodin(PHFile, ios::nocreate);
    if (!prodin)
    {
        cout<<"There is no Production History file"<<endl;
        exit(1);
    }
```

```
    prodin.getline(tempch,80);
    prodin>>numcol;
    for (i=1;i<=numcol;i++)
        prodin>>tempc;
    prodin>>numraw;
    for (i=1;i<=numraw;i++)
    {
        for (j=1;j<=numcol;j++)
        {
            prodin>>ProdHist[i][j];
        }
    }
    prodin.close();
    for (i=1; i<=numcol; i++)
    {
        sum = 0;
        for (j=1; j<=numraw; j++)
        {
            sum += ProdHist[j][i];
        }
        ProdHistVar[i] = sum/numraw;
        sum = 0;
        for (j=1; j<=numraw; j++)
        {
            sum += pow((ProdHist[j][i]-ProdHistVar[i]),2);
        }
        ProdHistVar[i] = sum/(numraw-1);
        if (ProdHistVar[i]<=1.0)
        {
            ProdHistVar[i] = 1;
        }
    }
    return;
}


//************** READ SIMULATED PRODUCTION FROM ECLIPSE RSM
FILE***************


void readSimProd()
{
    int i, j, jj;
    char tempch[150];
    double tempread[MAXHMCOL+2];
//
//  Read Input Parameters:
//
    ifstream simprodin("BASE.RSM", ios::nocreate);
    if (!simprodin)
    {
```

```
        cout<<"There is no Simulated Production History file"<<endl;
        exit(1);
    }
    else cout<<"File Found"<<endl;
    for (i=1;i<=6;i++)
        simprodin.getline(tempch,150);
    i = 0;
    while ((i<numraw)&&(!simprodin.eof()))
    {
        for (j=1;j<=(numcol+1);j++)
            simprodin>>tempread[j];
        if (tempread[1]==ProdHist[i+1][1])
        {
            i++;
            jj = 0;
            for (j=1;j<=(numcol+1);j++)
            {
                if (j!=2)
                {
                    jj++;
                    SimProd[i][jj] = tempread[j];
                }
            }
        }
    }
    if (i==0)
    {
        cout<<"Simulation for run = "<<RUNS<<" failed"<<endl;
    }
    simprodin.close();
    ofstream clearFilein("BASE.RSM", ios::trunc);
    clearFilein.close();
    return;
}


//******************** CALCULATE MISMATCH - OBJECTIVE FUNCTION **************


void ProdMatchError(int sol)
{
    int i, j;
    double sumcol, sumtot;
    sumtot = 0.0;
    for (i=2;i<=numcol;i++)
    {
        sumcol = 0.0;
        for (j=1;j<=numraw;j++)
        {
            sumcol += pow((SimProd[j][i]-ProdHist[j][i]),2)/ProdHistVar[i];
        }
```

```
        sumtot += sumcol;
    }
    objFunction[sol] = sumtot;
    return;
}


//************ DEKKER BRENT OPTIMIZATION PROCEDURE **********************


void DekkerBrent()
{
    int i, min, IterInner;
    double rdmin, rdint, rdmax, obfmin, obfint, obfmax, newrd, newobf;//, srd;
    bool change = false;
    SI = 0;
    for (i=2;i<=(nSavedEval-1);i++) {
        if ((objFunction[i]<=objFunction[i-1])&&(objFunction[i]<=objFunction[i+1])) {
            SI++;
            SIV[SI] = i;
        }
    }
    if (objFunction[1]<=objFunction[2]) {
        SI++;
        SIV[SI] = 1;
    }
    if (objFunction[nSavedEval]<=objFunction[nSavedEval-1]) {
        SI++;
        SIV[SI] = nSavedEval;
    }
    min = SIV[1];
    OBJ = objFunction[SIV[1]];
    for (i=2;i<=SI;i++) {
        if (objFunction[SIV[i]]<objFunction[min]) {
            min = SIV[i];
            OBJ = objFunction[SIV[i]];
        }
    }
    NEWOBJ = OBJ;
   IterInner = 0;
    while (IterInner<maxIterInnerLoop) {
        IterInner++;
        change = false;
        if (min==1) {
            rdv[3] = rdv[2];
            objFunction[3] = objFunction[2];
            rdv[2] = (2.0*rdv[1] + rdv[3])/3.0;
            rd = rdv[2];
            updatedsisim();
            system("$eclipse -file BASE");
            RUNS++;
```

```
            readSimProd();
            ProdMatchError(2);
            if (objFunction[2]<objFunction[1]) {
                min = 2;
                OBJ = objFunction[2];
                change = true;
            }
        }
        else {
            if (min==nSavedEval) {
                rdv[nSavedEval-2] = rdv[nSavedEval-1];
                objFunction[nSavedEval-2] = objFunction[nSavedEval-1];
                rdv[nSavedEval-1] = (2.0*rdv[nSavedEval] + rdv[nSavedEval-2])/3.0;
                rd = rdv[nSavedEval-1];
                updatedsisim();
                system("$eclipse -file BASE");
                RUNS++;
                readSimProd();
                ProdMatchError(nSavedEval-1);
                if (objFunction[nSavedEval-1]<objFunction[nSavedEval]) {
                    min = nSavedEval - 1;
                    OBJ = objFunction[nSavedEval-1];
                    change = true;
                }
            }
            else {
                rdmin = rdv[min-1];
                rdint = rdv[min];
                rdmax = rdv[min+1];
                obfmin = objFunction[min-1];
                obfint = objFunction[min];
                obfmax = objFunction[min+1];
                newrd = ((rdint+rdmax)*obfmin/((rdmin-rdint)*(rdmin-
rdmax))+(rdmin+rdmax)*obfint/((rdint-rdmin)*(rdint-rdmax))+(rdmin+rdint)*obfmax/((rdmax-
rdmin)*(rdmax-rdint)))/(2*(obfmin/((rdmin-rdint)*(rdmin-rdmax))+obfint/((rdint-rdmin)*(rdint-
rdmax))+obfmax/((rdmax-rdmin)*(rdmax-rdint))));
                if ((newrd<=1.0)&&(newrd>=0.0))
                    rd = newrd;
                else {
                    cout<<"Error in estimation of new rd: Check DekkerBrent (for constant values of
ObjFunct)";
                    rd = rd + (rdmax-rdint)/3.0;
                }
                updatedsisim();
                system("$eclipse -file BASE");
                RUNS++;
                readSimProd();
                ProdMatchError(0);
                newobf = objFunction[0];
                if (newrd<rdint) {
                    if (newobf<obfint) {
```

```
                    rdv[min] = newrd;
                    change = true;
                    objFunction[min] = newobf;
                    OBJ = newobf;
                    rdv[min+1] = rdint;
                    objFunction[min+1] = obfint;
                }
                else {
                    rdv[min-1] = newrd;
                    objFunction[min-1] = newobf;
                }
            }
            else {
                if (newobf<obfint) {
                    rdv[min] = newrd;
                    change = true;
                    objFunction[min] = newobf;
                    OBJ = newobf;
                    rdv[min-1] = rdint;
                    objFunction[min-1] = obfint;
                }
                else {
                    rdv[min+1] = newrd;
                    objFunction[min+1] = newobf;
                }
            }
        }
    }
    ofstream rdout2("RDout.txt", ios::ate);
    rdout2<<"     RD: "<<rdv[min]<<":   ObjFunction: "<<OBJ<<endl;
    rdout2.close();
    NEWOBJ = OBJ;
    ofstream runOut3("RunNumber.txt", ios::ate);
    runOut3<<"Run #"<<RUNS<<" (DB rd:"<<rdv[min]<<") - OBJFunction: "<<OBJ<<endl;
    runOut3.close();
    ofstream prodOut3("ProdOptResult.txt", ios::ate);
    prodOut3<<endl<<endl;
    prodOut3<<"DEKKER-BRENT, SimRun# "<<RUNS<<", ObjFunction: "<<OBJ<<endl;
    prodOut3<<"======================================================"<<endl;
    for (i=1;i<=numraw;i++)
    {
        for (j=1;j<=numcol;j++)
        {
            prodOut3<<"  "<<SimProd[i][j];
        }
        prodOut3<<endl;
    }
    prodOut3.close();
  }
  return;
}
```

```
//********************** UPDATE BEST CURRENT MODEL ************************


void updateClass()
{
   int ind, ic;
   for (ind=1;ind<=nxyz;ind++)
   {
      if (sim[ind]>UNEST)
      {
         clss[ind] = 0;
         for (ic=1;ic<=ncut;ic++)
         {
                  if (ivtype==0)
                  {
                       if (sim[ind]==thres[ic])
                 {
                            clss[ind] = ic;
                     }
                  }
                  else
                  {
                       if (sim[ind]<=thres[ic])
                       {
                            if (clss[ind]==0)
                                 clss[ind] = ic;
                     }
                  }
         }
       if (clss[ind]==0)
           clss[ind] = 99;
      }
   }
   return;
}
```

Appendix D: Input Files for History Match

Example of input parameter file:

```
VariableType 0
NumThresholds   5
ThresholdValues 1    2    3    4    5
PriorCDF/PDF    0.06    0.28    0.36    0.25    0.05
CondDataFile HardData.txt
CondFileDescrip 1    2    3    8
SoftDataFile   direct.ik
SoftFileDescrip   1    2    3    4    5    6    7    8
Markov-Bayes    0
Mark-BCalibBs   0.61    0.54    0.56    0.53    0.29
TrimmingLimits  -1.00E+211.00E+21
Min/MaxDataVal1    5
LowerTailOpt1    1
MiddleOption1    1
UpperTailOpt 1    5
TabulatedValFile cluster.dat
TabFileDescrip    4    0
DebuggingLevel  0
DebuggingFile    hmsisim.dbg
IncSimPermFile   SimRockType.INC
ProdHistFile   ProdHistory.txt
GridDescripX135     5776.14    101
GridDescripY 78  514.103    101
GridDescripZ 10  0.5 1.0
RandomSeedVal  740926
MaxCondDataKrig    10
MaxSimDataKrig10
MaxSoftDataKrig     1
AssignCondData  1
MultiGridOpt 0    3
NumDataPerOct  0
MaxSearchRadii  1200    700     2
SearchAngles 10  0    0
Full/MedianK 0    50
Simple/Ordinary  1
InitRealizOpt  0
InitRealizFile InitialRealiz.txt
SubdomainsOpt   0
SubdomainFile    Regions.dat
InitRDEvaluations     9
InnerLoopIter 3
OuterLoopIter      12
FirstVariogram    1    0.15
```

```
    1   0.85    10 0   0
    300     300     1
SecVariogram    1   0.15
    1   0.85    10 0   0
    400     400     1
ThirdVariogram  1   0.1
    1   0.85    10 0   0
    500     500     1
ForthVariogram  1   0.1
    1   0.85    10 0   0
    600     400     1
FifthVariogram  1   0.15
    1   0.85    10 0   0
    600     300     1
```

Parameter description for input file:

```
VariableType        =  1=continuous(cdf), 0=categorical(pdf)
NumThresholds       =  number thresholds/categories
ThresholdValues     =  thresholds / categories
PriorCDF/PDF        =  global cdf / pdf
CondDataFile        =  file with conditioning data
CondFileDescrip     =  columns for X,Y,Z, and variable
SoftDataFile        =  file with soft indicator input
SoftFileDescrip     =  columns for X,Y,Z, and indicators
Markov-Bayes        =  Markov-Bayes simulation (0=no,1=yes)
Mark-BCalibBs       =  calibration B(z) values
TrimmingLimits      =  trimming limits
Min/MaxDataVal      =  minimum and maximum data value
LowerTailOpt 1      =  lower tail option and parameter
MiddleOption 1      =  middle option and parameter
UpperTailOpt 1      =  upper tail option and parameter
TabulatedValFile    =  file with tabulated values
TabFileDescrip      =  columns for variable, weight
DebuggingLevel      =  debugging level: 0,1,2,3
DebuggingFile   =  file for debugging output
IncSimPermFile      =  file for simulation output
ProdHistFile        =  file with production history to be matched
GridDescripX        =  nx,xmn,xsiz
GridDescripY        =  ny,ymn,ysiz
GridDescripZ        =  nz,zmn,zsiz
RandomSeedVal       =  random number seed
MaxCondDataKrig  =  maximum original data  for each kriging
MaxSimDataKrig   =  maximum previous nodes for each kriging
MaxSoftDataKrig  =  maximum soft indicator nodes for kriging
```

AssignCondData = assign data to nodes? (0=no,1=yes)
MultiGridOpt = multiple grid search? (0=no,1=yes),num
NumDataPerOct = maximum per octant (0=not used)
MaxSearchRadii = maximum search radii
SearchAngles = angles for search ellipsoid
Full/MedianK = 0=full IK, 1=median approx. (cutoff)
Simple/Ordinary = 0=SK, 1=OK
InitRealizOpt = Initial Realization (0=Calculated Sisim; 1=From file)
InitRealizFile = File for initial realization
SubdomainsOpt = Number of Subdomains, followed by the region indexes
SubdomainFile = File with Subdomains or regions
InitRDEvaluations = Number of Initial RD Evaluations before DekkerBrent
InnerLoopIter = Number of inner loop DekkerBrent Iterations
OuterLoopIter = Number of outer loop Markov-Chain iterations
FirstVariogram = One nst, nugget effect
it,cc,ang1,ang2,ang3
a_hmax, a_hmin, a_vert
SecVariogram = Two nst, nugget effect
it,cc,ang1,ang2,ang3
a_hmax, a_hmin, a_vert

Example of conditioning data file:

```
Conditioning Data for Field Case
4
x
y
z
RockType
15348.96  6071.22   9.5    3
11466.07  6762.23   9.5    2
12781.03  3026.19   9.5    4
```

Example of production history file:

```
Production    History
9
TIME  FPR    FWCT WWC-P3 WWC-P4 WWC-P5 WWC-P11    WWC-P12    WWC-P13
4
0     3200   0            0  0  0  0  0  0  0
91    3150   0.0127432    0  0  0  0  0  0  0
182   3100   0.00906892   0  0  0  0  0  0  0
637   2850   0.325759     0  0  0  0  0  0  0
```

Appendix E: Upscaling Simulation File

Example of Eclipse® simulation files to upscale flow functions:

```
--RUNSPEC section----------------------------------------------------
RUNSPEC

TITLE
MODELBASE RUN FOR UPSCALING FLOW FUNCTIONS

--Request the LAB unit set
LAB

FMTOUT

UNIFOUT

--Oil water and gas are present
OIL
WATER
GAS

--IMPLICIT solution method
IMPLICIT

NSTACK
 -10 /

--Black oil
BLACKOIL

--Grid is 50 by 50 by 25 cells

DIMENS
50 50 25 /

TABDIMS
5 1 50 50 /

WELLDIMS
 5 300 5 5 /

START
  1 'JAN' 2007 /
```

```
--Grid section-------------------------------------------------------
GRID

--Basic grid block sizes
EQUALS
DX 4.0 /
DY 1.0 /
DZ 1.0 /
TOPS 222200 4* 1 1 /
/

INCLUDE
DiscPerm.INC /

INCLUDE
DiscPoro.INC /

COPY
PERMX PERMY 4* 1 25 /
PERMX PERMZ /
/

INIT

--Properties section-------------------------------------------------
PROPS

--Reservoir temperature
RTEMP
70    /

-- SURFACE DENSITIES OF RESERVOIR FLUIDS
--        OIL   WATER   GAS
DENSITY
    0.819121      0.999014      0.00084   /

--   PGAS   BGAS   VISGAS
PVDG
6.805  15.9    0.010
27.218 5.9  0.013
40.828 2.95    0.0135
54.437 1.96    0.014
68.046 1.47    0.0145
81.655 1.18    0.015
95.264 0.98    0.0155
108.873    0.84    0.016
122.483    0.74    0.0165
136.092    0.65    0.017
149.701    0.59    0.0175
163.31 0.54    0.018
176.919    0.49    0.0185
```

```
190.529    0.45    0.019
204.138    0.42    0.0195 /

--  RS      POIL    FVFO    VISO
--SCC/SCC      ATM    RCC/SCC    CPOISE
PVTO
  0.000       6.805   1.000    1.18  /
 29.388      27.218   1.012    1.17  /
 59.666      40.828   1.0255   1.14  /
 89.054      54.437   1.038    1.11  /
118.442      68.046   1.051    1.08  /
147.473      81.655   1.063    1.06  /
175.436      95.264   1.075    1.03  /
201.262     108.873   1.087    1.00  /
226.197     122.483   1.0985   0.98  /
247.570     136.092   1.11     0.95  /
267.161     149.701   1.12     0.94  /
284.972     163.310   1.13     0.92  /
298.508     176.919   1.14     0.91  /
311.688     190.529   1.148    0.9   /
322.375     204.138   1.155    0.89
          251.770   1.1504   0.89
          299.402   1.1458   0.89
          347.034   1.1412   0.89
          394.667   1.1367   0.89
          442.299   1.1321   0.89
          489.931   1.1275   0.89
          537.563   1.1230   0.89
          598.804   1.1184   0.89  /
/
--Water saturation functions
INCLUDE
SWFN.inc /

--Gas saturation functions
INCLUDE
SGFN.inc /

--Oil saturation functions
INCLUDE
SOF3.inc /

-- ROCK COMPRESSIBILITY
--    REF. PRES   COMPRESSIBILITY
ROCK
250   0.0000588 /

-- PVT PROPERTIES OF WATER
--    REF. PRES. REF. FVF  COMPRESSIBILITY  REF VISCOSITY  VISCOSIBILITY
PVTW
250 1.0 0.0000441 0.31 0.0 /
```

```
--Regions Section----------------------------------------------------
REGIONS

INCLUDE
SimRockType.INC /

--Solution section---------------------------------------------------
SOLUTION

--Request initial state output
RPTSOL
PRESSURE SOIL SWAT /

--OUTSOL
--PRESSURE SOIL /

-- DATA FOR INITIALISING FLUIDS TO POTENTIAL EQUILIBRIUM
--   DATUM DATUM  OWC   OWC   GOC   GOC   RSVD  RVVD  SOLN
--   DEPTH PRESS DEPTH  PCOW DEPTH  PCOG TABLE TABLE  METH
EQUIL
222200 340 274320 0 106680 0 1 1 0  /

-- VARIATION OF INITIAL RS WITH DEPTH
--   DEPTH   RS
RSVD
 30480  322.375
304800  322.375
/

SUMMARY  ============================================================

--Request items for summary files

RUNSUM

EXCEL
FPR
WOPR
'PRD'
/
WWCT
'PRD'
/
WBHP
'PRD'
'INJO'
'INJW'
/
FOSAT
FOVIS
```

```
FWVIS
FWSAT
BPR
1 25 2 /
1 25 5 /
1 25 8 /
1 25 11 /
1 25 14 /
1 25 17 /
1 25 20 /
1 25 23 /
50 25 2 /
50 25 5 /
50 25 8 /
50 25 11 /
50 25 14 /
50 25 17 /
50 25 20 /
50 25 23 /
/
BWPR
1 25 2 /
1 25 5 /
1 25 8 /
1 25 11 /
1 25 14 /
1 25 17 /
1 25 20 /
1 25 23 /
50 25 2 /
50 25 5 /
50 25 8 /
50 25 11 /
50 25 14 /
50 25 17 /
50 25 20 /
50 25 23 /
/

--Schedule section-------------------------------------------------------
SCHEDULE

TUNING
/
/
2* 100 1* 20 /

MESSAGES
 9* 100000 /
```

```
--Define injection and production wells
WELSPECS
'PRD' 'F' 50 25 222200 OIL /
'INJO' 'F' 1 25 222200 OIL /
'INJW' 'F' 1 25 222200 WATER /
/

COMPDAT
'PRD' 50 25 1 25 1* 1 1* 0.3/
'INJO' 1 1 8 8 1* 1 1* 0.3/
'INJO' 1 5 8 8 1* 1 1* 0.3/
'INJO' 1 9 8 8 1* 1 1* 0.3/
'INJO' 1 13 8 8 1* 1 1* 0.3/
'INJO' 1 17 8 8 1* 1 1* 0.3/
'INJO' 1 21 8 8 1* 1 1* 0.3/
'INJO' 1 25 8 8 1* 1 1* 0.3/
'INJO' 1 29 8 8 1* 1 1* 0.3/
'INJO' 1 33 8 8 1* 1 1* 0.3/
'INJO' 1 37 8 8 1* 1 1* 0.3/
'INJO' 1 41 8 8 1* 1 1* 0.3/
'INJO' 1 45 8 8 1* 1 1* 0.3/
'INJO' 1 49 8 8 1* 1 1* 0.3/
'INJW' 1 1 18 18 1* 1 1* 0.3/
'INJW' 1 5 18 18 1* 1 1* 0.3/
'INJW' 1 9 18 18 1* 1 1* 0.3/
'INJW' 1 13 18 18 1* 1 1* 0.3/
'INJW' 1 17 18 18 1* 1 1* 0.3/
'INJW' 1 21 18 18 1* 1 1* 0.3/
'INJW' 1 25 18 18 1* 1 1* 0.3/
'INJW' 1 29 18 18 1* 1 1* 0.3/
'INJW' 1 33 18 18 1* 1 1* 0.3/
'INJW' 1 37 18 18 1* 1 1* 0.3/
'INJW' 1 41 18 18 1* 1 1* 0.3/
'INJW' 1 45 18 18 1* 1 1* 0.3/
'INJW' 1 49 18 18 1* 1 1* 0.3/
/

WCONPROD
'PRD' OPEN BHP 5* 330 /
/

WCONINJE
--'INJO' 'OIL' 'OPEN' 'RATE' 20 4* 322.375 / --Live Oil
'INJO' 'OIL' 'OPEN' 'RATE' 0 2* / --Dead Oil
'INJW' 'WATER' 'OPEN' 'RATE' 20 2* /
/

--Solution output as requested in RPTSCHED
RPTSCHED
PRESSURE SOIL /
```

```
TSTEP
10000
/
END
```

Bibliography

Adler, P.M., Jacquin, C.J. and Quiblier, J.A.: "Flow in Simulated Porous-Media," *Intl. J. of Multiphase Flow* (1990) v.16, no.4, p.691.

Adler, P.M., Jacquin, C.J. and Thovert, J.-F.: "The formation factor of reconstructed porous media," *Water Resources Research* (1992) v.28, p.1571.

Al-Gharbi, M.S.: "Dynamic Pore-Scale Modeling of Two-Phase Flow," PhD Dissertation, Imperial College London, London, United Kingdom (2004).

Al-Gharbi, M.S. and Blunt, M.J.: "Dynamic Network Modeling of Two-Phase Drainage in Porous Media," *Physical Review E* (2005) v.71, p.23.

Al-Raoush, R.I.: "Extraction of Physically-Realistic Pore Network Properties from Three-Dimensional Synchrotron Microtomography Images of Unconsolidated Porous Media," PhD Dissertation, Louisiana State U., Baton Rouge, Louisiana (2002).

Amundsen, H. *et al*.: "Slow Two-Phase Flow in Artificial Fractures: Experiments and Simulations," *Water Resources Research* (1999) v.35, no.9, p.2619.

Archer, J.S. and Wong, S.W.: "Use of a Reservoir Simulator to Interpret Laboratory Waterflood Data," *SPEJ* (1973) v.12, p.343.

Baker, L.E.: "Three-Phase Relative Permeability Correlations," Paper SPE 17369, *Proc.*, 6th SPE/DOE Symposium on Enhanced Oil Recovery (1988), Tulsa, Oklahoma, 15-20 April.

Bakke, S. and Øren, P.-E.: "3-D Pore-Scale Modeling of Sandstones and Flow Simulations in the Pore Networks," *SPEJ* (1997) v.2, no.2, p.136.

Baldwin, C.A. *et al*.: "Determination and Characterization of the Structure of a Pore Space from 3D Volume Images," *J. of Colloid and Interface Science* (1996) v.181, p.79.

Bear, J.: "Dynamics of Fluids in Porous Media," Dover, New York City (1972).

Bernabe, Y.: "Pore Geometry and Pressure Dependence of the Transport Properties in Sandstones," *Geophysics* (1991) v.56, p.436.

Berryman, J.G.: "Measurement of Spatial Correlation Functions Using Image Processing Techniques," *J. of Applied Physics* (1985) v.57, no.7, p.2374.

Berryman, J.G.: "Relationship between Specific Surface Area and Spatial Correlation Function for Anisotropic Media," *J. of Math. Phys.* (1986) v.28, p.244.

Berryman, J.G. and Blair S.C.: "Use of Digital Image Analysis to Estimate Fluid Permeability of Porous Materials: Application of Two-point Correlation Functions," *J. of Applied Physics* (1986) v.60, p.1930.

Bird, B.B., Stewart, W.E. and Lightfoot, E.N.: "Transport Phenomena," (3rd ed.) John Wiley & Sons, Inc., New York City (1963).

Bissel, R.C., Sharma, Y. and Killough, J.E.: "History Matching Using the Method of Gradients: Two Case Studies", Paper SPE 28590 presented at the SPE 69th Annual Technical Conference and Exhibition (1994), New Orleans, LA.

Blanc, G., Hu, L.Y. and Noetinger, B.: "Gradual Deformation and Iterative Calibration of Sequential Stochastic Simulation", *Mathematical Geology* (2001) v.33, No.4, p.475.

Blair S.C., Berryman, J.G. and Berge, P.A.: "Using Two-point Correlation Functions to Characterize Microgeometry and Estimate Permeabilities of Sandstones and Porous Glass," *J. of Geophys. Res.* (1996) v.101, no.20, p.359.

Blunt, M.J.: "Effects of Heterogeneity and Wetting on Relative Permeability Using Pore Level Modeling," *SPEJ* (1997a) v.2, p.70.

Blunt, M.J.: "Flow in Porous Media: Pore-Network Models and Multiphase Flow," *Current Opinion in Colloids and Interface Science* (2001) v.6, no.3, p.197.

Blunt, M.J.: "Physically-Based Network Modeling of Multiphase Flow in Intermediate-Wet Porous Media," *J. of Petroleum Science and Engineering* (1998) v.20, p.117.

Blunt, M.J.: "Pore Level Modeling of the Effects of Wettability," *SPEJ* (1997b) v.2, p.494.

Blunt, M.J. *et al*.: "Detailed Physics, Predictive Capabilities and Macroscopic Consequences for Pore-Network Models of Multiphase Flow," *Advances in Water Resources* (2001) v.25, no.8, p.1069.

Blunt, M.J. and King, P.: "Macroscopic Parameters from Simulations of Pore Scale Flow," *Physical Review A* (1990) v.42, no.8, p.4780.

Blunt, M.J. and King, P.: "Relative Permeability from Two- and Three-Dimensional Pore-Scale Modeling," *Transport in Porous Media* (1991) v.6, p.407.

Blunt, M.J., King, P. and Scher, H.: "Simulation and Theory of Two-Phase Flow in Porous Media," *Physical Review A* (1992) v.46, p.7680.

Blunt, M.J., Zhou, D. and Fenwick, D.H.: "Three Phase Flow and Gravity Drainage in Porous Media," *Transport in Porous Media* (1995) v.20, p.77.

Bratvedt, F. Gimse, T. and Tegnander, C.: "Streamline Computations for Porous Media Fow Including Gravity," *Transport in Porous Media* (1996) v.25, no.1, p.63.

Bryant, S.P. and Blunt, M.J: "Prediction of Relative Permeability in Simple Porous-Media," *Physical Review A* (1992) v.46, no.4, p.2004.

Bryant, S.P., King, R. and Mellor, D.W.: "Network Model Evaluation of Permeability and Spatial Correlation in a Real Random Sphere Packing," *Transport in Porous Media* (1993) v.11, p.53.

Bryant, S.P., Mason, G. and Mellor, D.: "Quantification of Spatial Correlation in Porous Media and its Effect on Mercury Porosimetry," *J. of Colloid and Interface Science* (1996) v.177, p.88.

Burdine, N.T.: "Relative Permeability Calculation Size Distribution Data," *Pet. Trans. Am. Inst. Min. Metal. Pet. Eng.* (1953) v.198, p.71.

Caers, J.: "Geostatistical Reservoir Modeling Using Statistical Pattern Recognition," *J. of Petroleum Science and Engineering* (2001) v.29, no.3, p.177.

Caers, J.: "Markov Chain Theory for Spatial Stochastic Simulation," report no.12, Stanford Center for Reservoir Forecasting, Stanford U., Stanford, California, (May 1999) 1.

Caers, J.: "Methods for History Matching under Geological Constraints," Presented at the 8th European Conference on the Mathematics of Oil Recovery (2002), Freiburg, Germany, 3-6 September.

Cassel, D.K., Brown, J.M. and Johnson, G.A.: "Computer Tomographic Analysis of Water Distribution and Flow in Porous Media," *Theoretical and Applied Climatology* (1990) v.42, p.223.

Celia, M.A., Reeves, P.C. and Ferrand, L.A.: "Recent Advances in Pore Scale Models for Multiphase Flow in Porous Media," *Reviews of Geophysics* (1995) v.33, no.2, p.1049.

"Centre de Geostatistique," UNCERT, http://cg.ensmp.fr/index.html.

Chen, W.H. *et al*.: "A New Algorithm for Automatic History Matching," *SPEJ* (1974), p.593.

Childs, E.C. and Collis-George, N.: "The Permeability of Porous Materials," *Proceeding of the Royal Society of London* (1950) v.201, A, p.392.

Chiles, J.-P. and Delfiner, P.: "Geostatistics: Modeling Spatial Uncertainty," John Wiley & Sons, Inc., New York City (1999).

Chu, L., Reynolds, A.C., and Oliver, D.S.: "Computation of Sensitivity Coefficients for Conditioning the Permeability Field to Well-Test Pressure Data", *In Situ* (1995) v.19, No.2, p.179.

Cieplak, M., and Robbins, M.O.: "Influence of Contact Angle on Quasistatic Fluid Invasion of Porous Media," *Physical Review B* (1990) v.41, no.16, p.11508.

Coker, D.A. and Torquato, S.: "Extraction of Morphological Quantities from a Digitized Medium," *Journal of Applied Physics* (1995) v.77, p.955.

Craig, F.F.: "The Reservoir Engineering Aspects of Waterflooding," Monograph Series, SPE, Richardson, Texas (1971) v.3, p.12.

Datta-Gupta, A., Lake, L.W. and Pope, G.A.: "Characterizing Heterogeneous Permeable Media with Spatial Statistics and Tracer Data Using Sequential Simulated Annealing," *Mathematical Geology* (1995) v.27, no.6, p.763.

Deckman, H.W. *et al*.: "Microtomography Detector Design: It's not just Resolution," *Advances in X-ray Analysis* (1989) p.32.

Deckman, H.W. *et al*.: "Development of Quantitative X-ray Microtomography," *Matl. Res. Soc. Symp.* (1991) p.21.

Delshad, M. and Pope, G.A.: "Comparison of the Three-Phase Oil Relative Permeability Models," *Transport in Porous Media* (1989) v.4, p.59.

de Marsily, G. *et al*.: "Interpretation of Interference Tests in a Well Field Using Geostatistical Techniques to Fit the Permeability Distribution in a Reservoir Model," *In:* Geostatistics for Natural Resources Characterization, Part 2. (1984), p.831.

deMarsily, G. *et al*.: "Pilot Point Methodology for Automated Calibration of an Ensemble of Conditionally Simulation Transmissivity Fields: 2. Application", *Water Resources Research* (1995) v.31, No.3, p.495.

"Department of Applied Mathematics – Research," The Australian National U., http://wwwrsphysse.anu.edu.au/~mak110/disorder/disord2.html.

Deschamps, T. *et al*.: "The Results of Testing Six Different Gradient Optimizers on Two History Matching Problems," *Proc*., 6th European Conference on the Mathematics of Oil Recovery (1998) B-24, Peebles, UK.

Deutsch, C.V. and Journel, A.G.: "GSLIB: Geostatistical Software Library and Users Guide," (2nd ed.) Oxford University Press, New York City (1998).

Diaz, C.E., Chatzis, I. and Dullien, F.A.L.: "Simulation of Capillary Pressure Curves Using Bond Correlated Site Percolation on a Simple Cubic Network," *Transport in Porous Media* (1987) v.2, p.215.

Distefano, N. and Rath, A.: "An Identification Approach to Subsurface Hydrological Systems," *Water Resources Researc*h (1975) v.11, No.6, p.1005.

Dixit, A.B. *et al*.: "Pore-Scale Modelling of Wettability Effects and their Influence on Oil Recovery," Paper SPE 35451, *Proc.*, 10th SPE/DOE Symposium on Improved Oil Recovery (1996), Tulsa, Oklahoma, 17-20 April.

Dixit, A.B., McDougall, S.R. and Sorbie, K.S.: "Analysis of Relative Permeability Hysteresis Trends in Mixed-Wet Porous Media Using Network Models," Paper SPE 39656, *Proc.*, 11th SPE/DOE Symposium on Improved Oil Recovery (1998), Tulsa, Oklahoma, 19-22 April.

Dixit, A.B., McDougall, S.R. and Sorbie, K.S.: "A Pore-Level Investigation of Relative-Permeability Hysteresis in Water-Wet Systems," *SPEJ* (March 1998) p.115.

Doyen, P.M.: "Permeability, Conductivity, and Pore Geometry of Sandstone," *Journal of Geophysical Research* (1988) v.93, B7, p.7729.

Dullien, F.A.L.: "Porous Media: Fluid Transport and Pore Structure," Academic Press, Inc., New York City (1992).

Dullien, F.A.L.: "Porous Media, Fluid Transfer and Pore Structure," Academic Press, New York (1979).

Fenwick, D.H. and Blunt, M.J.: "Pore Level Modeling of Three Phase Flow in Porous Media," *Proc.*, 8th European Symposium on Improved Oil Recovery (1995), Vienna, Austria, 12-15 May.

Fenwick, D.H. and Blunt M.J.: "Three-Dimensional Modeling of Three-Phase Imbibition and Drainage," *Advances in Water Resources* (1998) v.21, no.2, p.121.

Ferrand, L.A. and Celia, M.A.: "The Effect of Heterogeneity on the Drainage Capillary Pressure-Saturation Relation," *Water Resources Research* (1992) v.28, no.3, p.859.

Fischer, U. and Celia, M.: "Prediction of Relative and Absolute Permeabilites for Gas and Water Retention Curves Using a Pore-Scale Network Model," *Water Resources Research* (1999) v.35, no.4, p.1089.

Flannery, B.P. *et al*.: "Three-Dimensional X-Ray Microtomography," *Science* (1987) v.237, p.1439.

Frandsen, P.E., Reffstrup, J. and Ansersen, J.: "History Matching Using the Multi Point Approximation Approach," *Proc*. 5th European Conference on the Mathematics of Oil Recovery (1996) p.295, Leoben, Austria.

Fredrich, J.T., Menendez, B. and Wong, T.-F.: "Imaging the Pore Structure of Geomaterials," *Science* (1995) v.268, p.276.

Gavalas, G.R., Shah, P.C. and Seinfeld, J.H.: "Reservoir History Matching by Bayesian Estimation," *SPEJ* (1976) v.16, no.6, p.337.

"Geohydrology Department Overview," Sandia National Laboratories, www.sandia.gov/ eesector/gs/gh/overview.html.

Glass, R.J. and Barrington, L.: "Simulation of Gravity Fingering in Porous Media Using a Modified Invasion Percolation Model," *Geoderma* (1996) v.70, p.231.

Glass, R.J., Nicholl, M.J. and Barrington, L.: "A Modified Invasion Percolation Model for Low-Capillary Number Immiscible Displacements in Horizontal Rough-Walled Fractures: Influence of Local in-Plane Curvature," *Water Resources Research* (1998) v.34, no.12, p.3215.

Goovaerts, P.: "Geostatistics for Natural Resources Evaluation," publ. Oxford Press, N.Y. (1997).

Gvirtzman, H. and Roberts, P.V.: "Pore Scale Spatial Analysis of Two Immiscible Fluids in Porous Media," *Water Resources Research* (1994) v.27, no.6, p.1165.

Hamon, G. and Roy, C.: "Influence of Heterogeneity, Wettability and Coreflood Design on Relative Permeability Curves," Paper SCA 2000-23, Society of Core Analysts (2004), Abu Dhabi.

Haugen, V.E., and Evensen, G.: "Assimilation of SLA and SST Data into OGCM for the Indian Ocean," *Ocean Dynamics* (2005) v.52, p.133.

Hazlett, R.D.: "Simulation of Capillary-Dominated Displacements in Microtomographic Images of Reservoir Rocks," *Transport in Porous Media* (1995) v.20, p.21.

Hazlett, R.D.: "Statistical Characterization and Stochastic Modeling of Pore Networks in Relation to Fluid Flow," *Mathematical Geology* (1997) v.29, no.6, p.801.

Held, R.J. and Celia, M.A.: "Modeling Support of Functional Relationships between Capillary Pressure, Saturation, Interfacial Area and Common Lines," *Advances in Water Resources* (2001) v.24, no.3, p.325.

Hilfer, R.: "Review on Scale Dependent Characterization of the Microstructure of Porous Media," *Transport in Porous Media* (2002) v.46, no.2, p.373.

Hilpert, M. and Miller, C.T.: "Pore-Morphology-Based Simulation of Drainage in Totally Wetting Porous Media," *Advances in Water Resources* (2001) v.24, p.243.

Hilpert, M., Glantz, R. and Miller, C.T.: "Calibration of a Pore-Network Model by a Pore-Morphological Analysis," Transport in Porous Media (2003) v.51, no.3, p.267.

Honarpour M. M., Cullick A. S. and Saad, N.: "Influence of Small-Scale Rock Laminations on Core Plug Oil/Water Relative Permeability and Capillary Pressure," SPE 27968, Centenial Petroleum Engineering Symposium (1994), Tulsa, OK.

Hu, L.Y.: "Combining Dependent Realizations within the Gradual Deformation Method," *Proc.*, 2001 Annual Conference of the International Association for Mathematical Geology, Cancun, Mexico 6-12 September.

Hu, L.Y.: "Gradual Deformation and Iterative Calibration of Gaussian-Related Stochastic Models," *Mathematical Geology* (2000) v.32, No.1, p.87.

Hu, L.Y. and Blanc, G.: "Constraining a Reservoir Facies Model to Dynamic Data Using a Gradual Deformation Model," Paper Presented at the 6th European Conference on Mathematics of Oil Recovery (1998), Peebles, Scotland, 8-11 September.

Hu, L.Y., Blanc, G. and Noetinger, B.: "Gradual Deformation and Iterative Calibration of Sequential Stochastic Simulations," *Mathematical Geology* (2001) 33, no.4, p.475.

Hu, L.Y. and Le Ravalec-Dupin, M.: "An Improved Gradual Deformation Method for Reconciling Random and Gradient Searches in Stochastic Optimizations," *Mathematical Geology* (2004) v.36, p.703.

Hughes, R.G. and Blunt, M.J.: "Network Modeling of Multiphase Flow in Fractures," *Advances in Water Resources* (2001) v.24, p.409.

Hughes, R.G. and Blunt, M.J.: "Network Modeling of Multiphase Flow in Fractures and Matrix/Fracture Transfer," Paper SPE 56411, *Proc.*, 1999 SPE Annual Technical Conference and Exhibition, Houston, Texas, 5-8 October.

Hughes, R.G. and Blunt, M.J.: "Pore Scale Modeling of Rate Effects in Imbibition," *Transport in Porous Media* (2000) v.40, no.3, p.295.

"Imperial College Consortium on Pore-Scale Modelling", Imperial College London, www3.imperial.ac.uk/portal/page?_pageid=46,68844&_dad=portallive&_schema=PORTALLIVE.

Ioannidis, M.A., Chatzis, I. and Sudicky, E.A.: "The Effect of Spatial Correlation on the Accessibility Characteristics of Three-Dimensional Cubic Networks as Related to Drainage Displacements in Porous Media," *Water Resources Research* (1993) v.29, no.6, p.1777.

Ioannidis, M.A. and Chatzis, I.: "Network Modeling of Pore Structure and Transport Properties of Porous Media," Pet. Trans. *Chemical Engineering Science* (1993b) v.48, no.5, p.951.

Ioannidis, M.A. and Chatzis, I.: "On the Geometry and Topology of 3D Stochastic Porous Media," *J. of Colloid and Interface Science* (2000) v.229, p.323.

Isichenko, M.B.: "Percolation, Statistical Topography, and Transport in Random Media," *Reviews of Modern Physics* (1992) v.64, p.961.

Jacquard, P. and Jain, C.: "Permeability Distribution from Field Pressure Data," *SPEJ* (1965) v.5, no.4, p.281.

Jahns, H.O.: "A Rapid Method for Obtaining a Two-Dimensional Reservoir Description from Well Pressure Response Data," *SPEJ* (1966) v.6, no.12, p.315.

Jerauld, G.R. and Salter, S.J.: "The Effect of Pore-Structure on Hysteresis in Relative Permeability and Capillary Pressure: Pore Level Modeling," *Transport in Porous Media* (1990) v.5, p.103.

Journel, A.G.: "Geostatistics: Models and Tools for the Earth Sciences," *Mathematical Geology* (1986) v.18, no.1, p.119.

Journel, A.G.: "Combining Knowledge from Diverse Sources: An alternative to Traditional Data Independence Hypotheses," *Mathematical Geology* (2002) v.34, No.5, p.573.

Kalaydjian, F.J.-M.: "Origin and Quantification of Coupling between Relative Permeabilities for Two-Phase Flows in Porous Media," *Transport in Porous Media* (1990) v.5, no.3, p.215.

Kalaydjian, F.J.-M.: "Performance and Analysis of Three-Phase Capillary Pressure Curves for Drainage and Imbibition in Porous Media," Paper SPE 24878, *Proc.*, 1992 SPE Annual Technical Conference and Exhibition, Washington, DC, 5-8 October.

Kalaydjian, F.J.-M. *et al*.: "Three-Phase Flow in Water-Wet Porous Media: Determination of Gas/Oil Relative Permeabilities Under Various Spreading Conditions," Paper SPE 26671, *Proc.*, 1993 SPE Annual Technical Conference and Exhibition, Houston, Texas, 5-8 October.

Kamath, J. *et al*.: "Use of Pore Network Models to Interpret Laboratory Experiments on Vugular Rocks," *J. Physical Sci*. (1998) v.20, p.109.

Kashib, T.: "A Probabilistic Approach to Dynamic Data Integration in Reservoir Models," MS Thesis, U. of Calgary, Calgary, Alberta, Canada (2003).

Kashib, T. and Srinivasan, S.: "Iterative Integration of Dynamic Data in Reservoir Models," Paper SPE 84592 Presented at the SPE Annual Technical Conference and Exhibition (2003), Denver, Colorado.

Kerig, P.D. and Watson, A.T.: "Relative-Permeability Estimation from Displacement Experiments: An Error Analysis," *SPERE* (1986) v.1, p.175.

King, P.R. *et al*.: "Applications of Statistical Physics to the Oil Industry: Predicting Oil Recovery Using Percolation Theory," *Physica A* (1999) v.274, p.60.

Kinney, J.H. and Nichols, M.C.: "X-Ray Tomographic Microscopy (XTM) Using Synchrotron Radiation," *Annu. Rev. Mater. Sci.* (1992) v.22, p.121.

Koplik, J., Lin, C. and Vermette, M.: "Conductivity and Permeability from Microgeometry," *Journal of Applied Physics* (1984) v.56, p.3127.

Kovscek, A.R., Wong, H. and Radke, C.J.: "A Pore-Level Scenario for the Development of Mixed Wettability in Oil Reservoirs," *AIChE J.* (1993) v.39, no.6, p.1072.

Krishnan, S., Boucher, A. and Journel, A.G.: "Evaluating information redundancy through the tau model," Proc. Geostatistics Congress (2004), Banff, Canada, Sep 26 – Oct 1.

Kulkami, K.N. and Datta-Gupta, A.: "Estimating Relative Permeability from Production Data: a Streamline Approach," *SPEJ* (2000) v.5, p.402.

Kwiecien, M.J., MacDonald, I.F. and Dullien, F.A.L.: "Three-Dimensional Reconstruction of Porous Media from Serial Section Data," *Journal of Microscopy* (1990) v.159, p.343.

Landa, J.L. and Horne, R.N.: "A Procedure to Integrate Well Test Data, Reservoir Performance History and 4-D Seismic Information into a Reservoir Description," Paper SPE 38653 presented at the SPE Annual Technical Conference and Exhibition (1997), San Antonio, TX.

Laroche, C. and Vizika, O.: "Two-Phase Flow Properties Prediction from Small-Scale Data Using Pore-Network Modeling," *Transport in Porous Media* (2005) v.61, no.1, p.77.

Lavenue, A.M. and Pickens, J.F.: "Application of a Coupled Adjoint Sensitivity and Kriging Approach to Calibrate a Groundwater Flow Model," *Water Resources Research* (1992) v.28, no.6, p.1543.

Le Ravalec-Dupin, M. and Noetinger, B.: "Optimization with the Gradual Deformation Method," *Mathematical Geology* (2002) v.34, no.2, p.125.

Lee, T.Y. and Seinfeld, J.H.: "Estimation of Two-Phase Petroleum Reservoir Properties by Regularization," *Journal of Computational Physics* (1987) v.69, p.397.

Lee, T.Y. and Seinfeld, J.H.: "Estimation of Absolute and Relative Permeability in Petroleum Reservoirs," *Inverse Problems* (1987) v.3, p.711.

Lenormand, R., Touboul, E. and Zarcone, C.: "Numerical Models and Experiments on Immiscible Displacements in Porous Media," *J. of Fluid Mechanics* (1988) v.189, p.165.

Lenormand, R., Zarcone, C. and Sarr, A.: "Mechanisms of the Displacement of One Fluid by Another in a Network of Capillary Ducts," *J. of Fluid Mechanics* (1983) v.135, p.337.

Li, R.: "Conditioning Geostatistical Models to Three-Dimensional Three-Phase Flow Production Data by Automatic History Matching," PhD Dissertation, U. of Tulsa, Tulsa, Oklahoma (2001).

Liang, Z.R. *et al*.: "Prediction of Permeability from the Skeleton of Three-Dimensional Pore Structure," *SPEREE* (1999) v.2, no.2, p.161.

Lin, C. and Cohen, M.H.: "Quantitative Methods for Microgeometric Modeling," *Journal of Applied Physics* (1982) v.53, p.4152.

Lindquist, W.B. *et al.*: "Medial Axis Analysis of Void Structure in Three Dimensional Tomographic Images of Porous Media," *Journal of Geophysical Research* (1996) v.101, B, p.8297.

Lindquist, W.B. and Venkatarangan, A.: "Investigating 3D Geometry of Porous Media from High Resolution Images," *Phys. Chem. Earth (A)* (1999) v.25, p.593.

Liu, N. and Oliver, D.S.: "Experimental Assessment of Gradual Deformation Method," *Mathematical Geology* (2004) v.36, no.1, p.65.

Lowry, M.I. and Miller, C.T.: "Pore-Scale Modeling of Nonwetting-Phase Residual in Porous Media," *Water Resources Research* (1995) v.31, no.3, p.455.

Ma, S., Mason, G. and Morrow, N.R.: "Effect of Contact Angle on Drainage and Imbibition in Regular Polygonal Tubes," *Colloids and Surfaces A* (1996) v.117, p.273.

MacDonald, I.F., Kaufman, P. and Dullien, F.A.L.: "Quantitative Image Analysis of Finite Porous Media, Development of Genus and Pore Map Software," *Journal of Microscopy* (1986a) v.144, p.277.

Makhlouf, E.M. *et al.*: "A General History Matching Algorithm for Three-Phase, Three-Dimensional Petroleum Reservoirs," *SPE Advanced Technology Series* (1993) v.1, no.2, p.83.

Makse, H.A. *et al.*: "Pattern Formation in Sedimentary Rocks: Connectivity, Permeability and Spatial Correlations," *Physica A* (1996) v.233, p.587.

Man, H.N. and Jing X.D.: "Pore Network Modeling of Electrical Resistivity and Capillary Pressure Characteristics," *Transport in Porous Media* (2000) v.41, p.263.

Mani, V. and Mohanty, K.K.: "Effect of Pore-Space Spatial Correlations on Two-Phase Flow in Porous Media," *J. of Petroleum Science and Engineering* (1999) v.23, p.173.

Mannseth T. *et al.*: "Effects of Rock Heterogeneities on Capillary Pressure and Relative Permeabilities," SPE 50574, European Petroleum Conference (1998), The Hague, The Netherlands.

Manwart, C., Torquato, S. and Hilfer, R.: "Stochastic Reconstruction of Sandstones," *Physical Review E* (2000) v.62, no.1, p.893.

Margulis, S.A., *et al*.: "Land Data Assimilation and Estimation of Soil Moisture Using Measurements from the Southern Great Plains 1997 Field Experiment," *Water Resources Research* (2002) v.38, p.12.

Matthews, G.P., Moss, A.K. and Ridgeway, C.J.: "The Effects of Correlated Networks on Mercury Intrusion Simulations and Permeabilities of Sandstone and Other Porous Media," *Powder Technology* (1995) v.83, p.61.

McDougall, S.R. and Sorbie, K.S.: "The Impact of Wettability on Waterflooding: Pore-Scale Simulation," *SPERE* (1995) v.10, p.208.

Mezghani, M. and Roggero, F.: "Combining Gradual Deformation and Upscaling Techniques for Direct Conditioning of Fine Scale Reservoir Models to Dynamic Data," Paper SPE 71334 Presented at the 2001 SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 1-3 October.

Mijare, O. *et al*.: "Numerical Simulation Model of Highly Faulted Reservoir Based on an Integrated Approach in Lake Maracaibo," Paper SPE 69579 Presented at the SPE Latin America and Caribbean Petroleum Engineering Conference (2001) Buenos Aires, Argentina, 25-28 March.

Miller, C.T. *et al*.: "Multiphase Flow and Transport Modeling in Heterogeneous Porous Media: Challenges and Approaches," *Advances in Water Resources* (1998) v.21, no.2, p.77.

Mogensen, K. and Stenby, E.: "A Dynamic Pore-Scale Model of Imbibition," Paper SPE 39658, Proc., 11th SPE/DOE Symposium on Improved Oil Recovery (1998), Tulsa, Oklahoma, 19-22 April.

Mohanty, K.K. and Salter, S.J.: "Multiphase Flow in Porous Media: II Pore-Level Modeling," Paper SPE 11018 Presented at the 1982 SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 26-29 September.

Morris, J.P., Zhu, Y. and Fox, P.J.: "Parallel Simulations of Pore-Scale Flow Through Porous Media," *Computers and Geotechnics* (1999) v.25, no.4, p.227.

Mualem, Y.: "A New Model for Predicting the Hydraulic Conductivity of Unsaturated Porous Media," *Water Resources Research* (1976) v.12, no.3, p.513.

Naevdal, G., *et al*.: "Reservoir Monitoring and Continuous Model Updating Using Ensemble Kalman Filter," Paper SPE 84372 presented at the SPE Annual Technical Conference and Exhibition (2003), Denver, CO.

Nilsen, L.S. *et al*.: "Prediction of Relative Permeability and Capillary Pressure from a Pore Model," Paper SPE 35531, *Proc.*, 1996 European 3-D Reservoir Modeling Conference, Stavanger, Norway, 16-17 April.

Oliver, D.S.: "Multiple Realizations of the Permeability Field from Well-Test Data," *SPEJ* (1996) v.1, no.2, p.145.

Oliver, D.S.: "On Conditional Simulation to Inaccurate Data," *Mathematical Geology* (1996) v.28, no.6, p.811.

Oliver D.S. *et al*.: "Integration of Production Data into Reservoir Models," *Petroleum Geoscience* (2001) v.7, p.65.

Øren, P.-E. and Bakke, S.: "Process Based Reconstruction of Sandstones and Prediction of Transport Properties," *Transport in Porous Media* (2002) v.46, No.2, p.311.

Øren, P.-E., Bakke, S. and Artzen, O.J.: "Extending Predictive Capabilities to Network Models," Paper SPE 38880, *Proc.*, 1997 SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 5-8 October.

Øren, P.-E., Billiote, J. and Pinczewski, W.V.: "Pore-Scale Network Modeling of Waterflood Residual Oil Recovery by Immiscible Gas Flooding," Paper SPE 27814, *Proc.*, 9th SPE/DOE Symposium on Improved Oil Recovery (1994), Tulsa, Oklahoma, 17-20 April

Pan, C.: "Use of Pore-Scale Modeling to Understand Flow and Transport in Porous Media," PhD Dissertation, U. of North Carolina at Chapel Hill, Chapel Hill, North Carolina (2003).

Pan, C., Hilpert, M. and Miller, C.T.: "Pore-Scale Modeling of Saturated Permeabilities in Random Sphere Packings," *Physical Review E* (2001) v.64, p.20.

Paterson, L. *et al*.: "Simulating Residual Saturation and Relative Permeability in Heterogeneous Formations," Paper SPE 36523 Presented at the 1996 SPE Annual Technical Conference and Exhibition, Denver, Colorado, 6-9 October.

Patzek, T.W.: "Verification of a Complete Pore Network Simulator of Drainage and Imbibition," *SPEJ* (2001) v.6, no.2, p.144.

Pereira, G.G. *et al*.: "Pore-Scale Network Model for Drainage Dominated Three-Phase Flow in Porous Media," *Transport in Porous Media* (1996) v.24, no.2, p.157.

Piri, M.: "Pore-Scale Modeling of Three-Phase Flow," PhD Dissertation, Imperial College London, London, United Kingdom (2003).

Piri, M. and Blunt M.J.: "Three-Dimensional Mixed-Wet Random Pore-Scale Network Modeling of Two- and Three-Phase Flow in Porous Media," *Physical Review E* (2005) v.71, p.253.

"Pore Scale Modeling," The Geoanalysis Group (Ees-5), http://ees-www.lanl.gov/EES5/ models/examples/porescale.html.

Quiblier, J.A.: "A New Three-Dimensional Modeling Technique for Studying Porous Media," *J. Colloid. Interface Sci.* (1948) v.98, p.84.

Rajaram, H., Ferrand, L.A. and Celia, M.A.: "Prediction of Relative Permeabilities for Unconsolidated Soils Using Pore-Scale Network Models," *Water Resources Research* (1997) v.33, no.1, p.43.

Reynolds, A.C., Li, R. and Oliver, D.S.: "Simultaneous Estimation of Absolute and Relative Permeability by Automatic History Matching of Three-Phase Flow Production Data," Paper 2001-07 Presented at the 2001 Petroleum Society's Canadian International Petroleum Conference, Calgary, Alberta, Canada, 12-14 June.

Roggero, F. and Hu, L.Y.: "Gradual Deformation of Continuous Geostatistical Models for History Matching," Paper SPE 49004, presented at the SPE Annual Technical Conference and Exhibition (1998), New Orleans, LA.

Sahimi, M.: "Long-Range Correlated Percolation and Flow and Transport in Heterogeneous Porous Media," *J. de Physique I France* (1994) v.4, p.1263.

Sahimi, M.: "Effect of Long-Range Correlations on Transport Phenomena in Disordered Media," *AIChE J.* (1995) v.41, p.229.

Schlueter, E.M. et al.: "The Fractal Dimension of Pores in Sedimentary Rocks and its Influence on Permeability," *Engineering Geology* (1997) v.48, p.199.

Sigmund, P.M. and McCaffery, FG.: "An Improved Unsteady-State Procedure for Determining the Relative Permeability Characteristics of Heterogeneous Porous Media," *SPEJ* (1979) v.19, p.15.

Soll, W.E. and Celia, M.A.: "A modified Percolation Approach to Simulating Three-Fluid Capillary Pressure-Saturation Relationships," *Advances in Water Resources* (1993) v.16, no.2, p.107.

Soll, W.E., Celia, M.A. and Willson, J.L.: "Micromodel Studies of Three-Fluid Porous Media Systems: Pore-Scale Processes Relating to Capillary Pressure-Saturation Relationships," *Water Resources Research* (1993) v.29, no.9, p.2963.

Spanne, P. and Rivers, M.L.: "Computerized Microtomography Using Synchrotron Radiation from the Nsls," *Nucl. Instrum. Methods Phys. Res.* (1987) v.25, B24, p.1063.

Srinivasan, S. and Caers, J.: "Conditioning Reservoir Models to Dynamic Data – A Forward Modeling Perspective," Paper SPE 62941 Presented at the 2000 SPE Annual Technical Conference and Exhibition, Dallas, Texas, 1-4 October.

Srinivasan, S. and Bryant S.: "Integrating Dynamic Data in Reservoir Models Using a Parallel Computational Approach," Paper SPE 89444 Presented at the 2004 SPE/DOE 13th Symposium on Improved Oil Recovery, Tulsa, Oklahoma, 17-21 April.

"Stanford Center for Reservoir Forecasting," Stanford U., http://corporate.stanford.edu/ research/programs/reserv.html.

Strebelle, S.: "Conditional Simulation of Complex Geological Structures Using Multiple-Point Statistics," *Math. Geology* (2002) v.34, no.1, p.1.

Tarantola, A.: "Inverse Problem Theory: Methods for Data Filling and Model Parameter Estimation," Elsevier (1987), Amsterdam.

Thompson, K.E. and Fogler, H.S.: "Modeling Flow in Disordered Packed Beds from Pore-Scale Fluid Mechanics," *AIChE J.* (1997) v.43, p.1377.

Thovert, J.-F., Salles, J. and Adler, P.M.: "Computerized Characterization of the Geometry of Real Porous Media: their Discretization, Analysis and Interpretation," *Journal of Microscopy* (1993) v.170, p.65.

Tiab, D. and Donaldson, E.: "Petrophysics: Theory and Practice of Measuring Reservoir Rock and Fluid Transport Properties," Gulf Publishing House, Houston, Texas (1996).

Valvatne, P.H. *et al*.: "Predictive Pore-Scale Modeling of Single and Multiphase Flow," *Transport in Porous Media* (2005) v.58, no.1-2, p.23.

Valvatne, P.H. and Blunt, M.J.: "Predictive Pore-Scale Modeling of Two-Phase Flow in Mixed Wet Media," *Water Resources Research* (2004) v.40, p.321.

Venkatarangan, A.B.: "Geometric and Statistical Analysis of Porous Media," PhD Dissertation, State U. of New York, Stony Brook, New York (2000).

Vogel H.J.: "A Numerical Experiment on Pore Size, Pore Connectivity, Water Retention, Permeability and Solute Transport Using Network Models," *European J. of Soil Science* (2000) v.51, p.99.

Vogel, H.J. and Roth, K.: "Quantitative Morphology and Network Representation of Soil Pore Structure," *Advances in Water Resources* (2001) v.24, p.233.

Wardlaw, N.C., Li, Y. and Forbes, D.: "Pore-Throat Size Correlation from Capillary Pressure Curves," *Transport in Porous Media* (1987) v.2, p.597.

Wardlaw, N.C. and Taylor, R.P.: "Mercury Capillary Pressure Curves and the Interpretation of Pore Structure and Capillary Behavior of Reservoir Rocks," *Bull. Can. Petro. Geol.* (1976) v.24, p.225.

Watson, A.T. *et al*.: History Matching in Two Phase Petroleum Reservoirs. *SPEJ* (1980) v.20, p.521.

Weibull,W.: "A Statistical Distribution Function of Wide Applicability," *Journal of Applied Mechanics* (1951) v.18, p.293.

Wong, T.-F. and Zhu, W.: "Brittle Faulting and Permeability Evolution: Hydromechanical Measurement, Microstructural Observation, and Network Modeling, in Faults and Subsurface Fluid Flow," *American Geophysical Union*, Washington, DC (1999).

Wu, Z., Reynolds, A.C. and Oliver, D.S.: "Conditioning Geostatistical Models to Two-Phase Production Data," Paper SPE 56885 Presented at 1998 SPE Annual Technical Conference and Exhibition, New Orleans, Louisiana, 27-30 September.

Xue, G. and Datta-Gupta, A.: "Structure Preserving Inversion: An Efficient Approach to Conditioning Stochastic Reservoir Models to Dynamic Data," Paper SPE 38727 Presented at 1997 SPE Annual Technical Conference and Exhibition, San Antonio, Texas, 5-8 October.

Yale, D.P.: "Network Modeling of Flow, Storage and Deformation in Porous Rocks," PhD Dissertation, Stanford U., Stanford, California (1984).

Yang, P.-H. and Watson, A.T.: "Automatic History Matching with Variable-Metric Methods," *SPERE* (1988) v.3, no.3, p.995.

Yang, P.-H. and Watson, A.T: "A Bayesian Methodology for Estimating Relative Permeability Curves," *SPERE* (1991) v.6, p.259.

Yanuka, M., Dullien, F.A.L. and Elrick, D.E.: "Serial Sectioning and Digitization of Porous Media for Two- and Three-Dimensional Analysis and Reconstruction," *Journal of Microscopy* (1984) v.135, p.159.

Yeong, C.L.Y. and Torquato, S.: "Reconstructing Random Media," *Physical Review E* (1998a) v.57, No.1, p.495.

Yortsos, Y.C. *et al*.: "Use of Pore-Network Models to Simulate Laboratory Corefloods in a Heterogeneous Carbonate Sample," *SPEJ* (1999) v.4, p.179.

Zhang, F. and Reynolds, A.C.: "Optimization Algorithms for Automatic History Matching of Production Data," Presented at the 8th European Conference on the Mathematics of Oil Recovery (2002), Freiburg, Germany, 3-6 September.

Zhang, F. *et al*.: "Automatic History Matching in a Bayesian Framework, Example Applications," Paper SPE 84461 Presented at the 2003 SPE Annual Technical Conference and Exhibition, Denver, Colorado, 5-8 October.

Vita

Alvaro Enrique Barrera was born in Bucaramanga, Colombia on September 26, 1974, the son of Alvaro Barrera Larrarte and Eugenia Gomez de Barrera. After completing his high school education at Colegio San Pedro Claver, Bucaramanga, in 1992, he entered the Universidad Industrial de Santander in Bucaramanga. He received the degree of Bachelor of Science in Petroleum Engineering in 1999. During the following three years he worked on research as a petroleum engineer at the Centro de Investigación del Gas from the Universidad Industrial de Santander and at the Instituto de Investigación del Petróleo, ICP, research institute from the Empresa Colombiana de Petróleos, ECOPETROL. In September 2002, he entered The Graduate School at The University of Texas at Austin.

Permanent Address: 1421 Charolais Dr.
Austin, Texas 78758

This dissertation was typed by the author.